

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS

RAFAEL SUETSUGU MELLO

**SEUHORTIFRUTI:
SISTEMA WEB PARA *DELIVERY* DE PRODUTOS ALIMENTÍCIOS**

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

PATO BRANCO
2021

RAFAEL SUETSUGU MELLO

**SEUHORTIFRUTI:
SISTEMA WEB PARA *DELIVERY* DE PRODUTOS ALIMENTÍCIOS**

SEUHORTIFRUTI: WEB SYSTEM FOR GROCERY DELIVERY

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Dra. Eliane De Bortoli Fávero

PATO BRANCO
2021



Esta licença permite que outros remixem, adaptem e criem a partir do trabalho licenciado para fins não comerciais, desde que atribuam ao autor o devido crédito e que licenciem as novas criações sob termos idênticos.

RAFAEL SUETSUGU MELLO

**SEUHORTIFRUTI:
SISTEMA WEB PARA *DELIVERY* DE PRODUTOS ALIMENTÍCIOS**

Trabalho de Conclusão de Curso de Graduação
apresentado como requisito para obtenção do título de
Tecnólogo em Análise e Desenvolvimento de Sistemas da
Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 06/dezembro/2021

Prof. Dr. Marcelo Teixeira
Universidade Tecnológica Federal do Paraná

Profa. Msc. Andreia Scariot Beulke
Universidade Tecnológica Federal do Paraná

Profa. Dra. Eliane De Bortoli Fávero
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2021

RESUMO

Devido a pandemia de *Covid-19*, o serviço de entrega de alimentos a domicílio, também conhecido como *delivery* se tornou mais evidente no cotidiano das empresas, em especial no setor de varejo de alimentos, que conta com alta rotatividade no estoque, tendo o alcance limitado ao seu bairro de atuação devido a lockdowns e medidas restritivas. Os serviços de *delivery* passaram a ser uma necessidade para empresas que desejam manter seu negócio ativo nesse ramo de atuação. Para empresas desse setor, ter um aplicativo próprio apresenta benefícios, tanto para o empreendedor quanto seus clientes, oferecendo comodidade e praticidade para o cliente e maior abrangência de mercado consumidor para o estabelecimento. Sendo assim, o presente trabalho apresenta a implementação de um aplicação web responsável (*PWA*) para *delivery* de alimentos orgânicos que possibilite à pequenas e médias empresas o cadastro e a comercialização de seus produtos pela Internet.

Palavras-chave: *Delivery* de hortigranjeiros. *E-commerce*. Hortigranjeiros. Varejo de alimentos. Comércio eletrônico.

ABSTRACT

Due to the *Covid-19* pandemic, the delivery service has become more evident in the daily lives of companies, especially in the food retail sector, which has a high stock turnover, having limited reach to its neighborhood due to lockdowns and restrictive measures. Home delivery services have become a necessity for companies that wish to have space in the market. For companies in this sector, having an own application presents benefits, both for the entrepreneur and his clients, offering convenience and practicality for the client and greater scope of the consumer market for the establishment. Therefore, the present work present the implementation of a responsible web application (PWA) for delivery of organic foods that allows small and medium-sized companies to register and sell their products over the Internet.

Keywords: *Delivery. E-commerce. Horticulture. Food Delivery. E-commerce.*

LISTA DE FIGURAS

Figura 1 – Tela inicial do e-commerce <i>delivery</i> hortifruti.....	18
Figura 2 -Tela inicial da aplicação <i>frutello delivery</i>	18
Figura 3 – Tela inicial do e-commerce <i>Organicos In Box</i>	19
Figura 4 – Página de hortifruti no marketplace <i>Ifood</i>	20
Figura 5 – Diagrama de caso de uso	29
Figura 6 – Diagrama sequencial de adicionar no carrinho	34
Figura 7 – Diagrama de sequência para efetuar pedido.....	34
Figura 8 – Diagrama de sequência para consultar pedidos	35
Figura 9 – Diagrama de sequência para alterar informações da empresa	35
Figura 10 – Diagrama de atividades para o processo de Efetuar Pedido	36
Figura 11 – Diagrama de classes preliminar	37
Figura 12 – Diagrama de entidade relacionamento	38
Figura 13 – Página inicial do aplicativo	40
Figura 14 – Responsividade da página inicial	43
Figura 15 – Tela de login do usuário	43
Figura 16 – Tela de cadastro de usuário	44
Figura 17 – Painel do administrador	46
Figura 18 – Modal para cadastro de produto.....	48
Figura 19 – Tela do carrinho de compras	50
Figura 20 – Tela perfil do usuário	52
Figura 21 – Tela do <i>whatsapp</i> após a finalização do pedido	54
Figura 22 – Tela dados da empresa	56
Figura 23 – Tela sobre nós	57
Figura 24 – Instalação PWA	58
Figura 25 – Relatório do Google Lighthouse	61

LISTA DE QUADROS

Quadro 1 – Lista de ferramentas e tecnologias	22
Quadro 2 – Requisito Cadastrar Produto	26
Quadro 3 – Requisito Listar Produtos	26
Quadro 4 – Requisito Controlar carrinho de compras	26
Quadro 5 – Requisito Informar endereços de entrega	27
Quadro 6 – Requisito Escolher horário de entrega	27
Quadro 7 – Requisito Calcular valor aproximado da compra	27
Quadro 8 – Requisito Registrar pedido	28
Quadro 9 – Requisito Listar pedidos	28
Quadro 10 – Requisito Cadastrar usuários.....	29
Quadro 11 – Requisito Cadastrar dados da empresa	29
Quadro 12 – Caso de uso Cadastrar usuário	30
Quadro 13 – Caso de uso Logar no sistema	30
Quadro 14 – Caso de uso adicionar produto no carrinho de compras	31
Quadro 15 – Caso de uso Efetuar pedido	31
Quadro 16 – Caso de uso Consultar pedidos.....	32
Quadro 17 – Caso de uso Gerenciar produtos	33
Quadro 18 – Caso de uso Alterar dados da empresa	33
Quadro 19 – Caso de uso Alterar status dos pedidos	34
Quadro 20 – Cadastrar status dos pedidos	34
Quadro 21 – Caso de uso Gerar relatórios de vendas	34

LISTAGENS DE CÓDIGO

Listagem 1 - Rota para tela principal	41
Listagem 2 - Página inicial apresenta produtos	42
Listagem 3 - Autenticação de usuários	43
Listagem 4 - Cadastro de usuários	45
Listagem 5 - Função que retorna o painel do administrador	46
Listagem 6 - Painel do administrador	47
Listagem 7 - Cadastro de produtos	49
Listagem 8 - Controle do carrinho de compras	51
Listagem 9 - Controle do carrinho de compras	52
Listagem 10 - Controle do carrinho de compras	55
Listagem 11 - Arquivo ServiceWorker.js	59
Listagem 12 - Arquivo Manifest.json	60

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de programação de aplicação
APP	Aplicativo
CEP	Código de Endereçamento Postal
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
PWA	Progressive Web Application
RF	Requisito Funcional
RNF	Requisito Não Funcional
UI	Interface de usuário
URL	Uniform Resource Locator
UX	Experiência de usuário
WEB	Nome usado para designar a própria Internet

SUMÁRIO

1 INTRODUÇÃO	8
1.1 CONSIDERAÇÕES INICIAIS	8
1.2 OBJETIVOS	10
1.2.1 Objetivo Geral	12
1.2.2 Objetivos Específicos	12
1.3 JUSTIFICATIVA	12
1.4 ESTRUTURA DO TRABALHO	13
2 REFERENCIAL TEÓRICO	14
2.1 E-COMMERCE	14
2.2 DELIVERY EM HORTIGRANJEIROS	16
2.3 PROGRESSIVE WEB APPS (PWA)	17
2.4 SISTEMAS HORTIFRUTIGRANJEIROS EXISTENTES	18
3 MATERIAIS E MÉTODO.....	17
3.1 MATERIAIS.....	22
3.2 MÉTODO	23
4 RESULTADOS.....	25
4.1 ESCOPO DO SISTEMA.....	25
4.2 MODELAGEM DO SISTEMA.....	25
4.3 APRESENTAÇÃO DO SISTEMA.....	39
4.4 CONSIDERAÇÕES FINAIS.....	49
REFERÊNCIAS.....	50

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa da realização deste trabalho. No final do capítulo está a organização do texto por meio de uma breve apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

Com a crise da *Covid-19*, o serviço de entrega de produtos alimentícios à domicílio, também conhecido como *delivery* de alimentos, passou de um diferencial para uma necessidade do mercado. Na retomada da economia, mais da metade (53%) dos donos de bares e restaurantes relatam que suas empresas operam no prejuízo, e para 52% deles o faturamento está abaixo da expectativa (ABRASEL, 2020).

Uma forma de mitigar os impactos sofridos com a pandemia pelo setor, vem sendo o fortalecimento ou a implantação do *delivery*. O serviço de entrega tem sido a única forma de muitos estabelecimentos de produtos alimentícios continuarem operando (MARTINS, 2020).

“Os empresários que enxergaram oportunidades e se reinventaram saltaram na frente dos outros. Segundo nossa pesquisa, entre as micro e pequenas empresas que tiveram crescimento na receita durante a crise, cerca de 48% mudaram o modo de funcionamento, apostando mais no atendimento de solicitações de entregas *on-line* e serviços na *Internet*” (MELLES, 2020).

O serviço de entrega *on-line* está crescendo rapidamente, reunindo inovação e conveniência aos seus clientes, juntamente com facilidade de acesso a telefones celulares e à Internet, que contribuíram para o surgimento de um grande número de plataformas que oferecem este serviço no Brasil (PIGATTO; MACHADO; NEGRETI, 2017). Percebe-se que, por prover conforto e facilidade, a tendência é que cada vez mais pessoas experimentem essa maneira prática de receber alimento em casa. Da mesma forma, muitos empresários passaram a se interessar na oportunidade da abertura de novos canais de venda, seja por meio de *delivery* ou *e-commerce*.

A conveniência e as funcionalidades oferecidas pelos dispositivos móveis, tais como telefones celulares, tiveram como resultado muitas pessoas optam pelo uso da mobilidade. Dois dos maiores grupos de pessoas que utilizam dispositivos móveis são os profissionais e os consumidores (LEE; SHINEIDER; SCHELL, 2005, p. 10).

Atualmente, *e-marketplace* é um modelo de negócio *on-line* baseado na intermediação entre compradores e vendedores que possibilitam a comercialização por meio de sua plataforma (CAMPOS et al., 2015). Ter seu próprio aplicativo oferece vantagens que atraem empreendedores, tais como: capacidade de personalização do *layout* do *site*, domínio próprio, dispensabilidade de taxas de comissões por venda, ações de relacionamento com o cliente, etc. Considerando a atual conjuntura, verifica-se que existem inúmeros benefícios para essas aplicações próprias ainda que o mercado de aplicativos móveis seja dominado pelos modelos de *marketplaces*. Segundo Sergio Molinari (2020), os *marketplaces* dominam a maior parte dos pedidos por *delivery* com 61%, seguido por pedidos via telefone do estabelecimento com 34%. O autor ressalta a relevância dos pedidos por *Whatsapp*, com uma fatia de 32% do total.

Para alcançar todos os dispositivos é preciso projetar *sites* responsivos, pois esse é o futuro da *web* (FRANÇA, 2015). Levando em consideração a demanda das empresas e crescimento do mercado de *delivery* via aplicativos móveis, o presente trabalho visa construir uma *Progressive Web Application* (PWA) ou um sistema *web* PWA para *delivery* de alimentos orgânicos que possibilite para as pequenas e médias empresas o cadastro e a comercialização de seus produtos. Os PWAs são um conjunto de estratégias, técnicas e *API* que permitem que os desenvolvedores forneçam aos usuários o tipo de experiência *mobile* nativa do dispositivo móvel que eles estão acostumados (SHEPPARD, 2017), trazendo os recursos que se espera dos aplicativos nativos para o navegador móvel. De forma que ele use tecnologias baseadas em padrões e seja executado em um contêiner seguro que é acessível a qualquer pessoa na *web* (TANDEL, S.; JAMADAR, 2018).

A proposta surge como forma de suprir uma demanda real e existente em um estabelecimento hortifrutigranjeiro. A solução de desenvolver uma aplicação para facilitar o processo de comercialização de produtos alimentícios surgiu como forma de mitigar o problema de atendimentos de *delivery*, o que vem sendo realizado via *Whatsapp*.

Tratando-se de um novo canal de vendas, a aplicação dará ao proprietário a possibilidade de ter seus produtos expostos para um público muito mais abrangente, graças ao alcance da Internet, corroborando para o aumento de seu faturamento. Assim, o estabelecimento tem a possibilidade de atingir um público *on-line* bastante diverso, tanto por meio do aplicativo *mobile*, quanto por meio de navegadores em computadores *desktop*. A aplicação proposta e desenvolvida neste trabalho de conclusão de curso visa auxiliar hortigranjeiros e mercearias do ramo mediante o atendimento dos objetivos especificados no tópico a seguir.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver uma PWA para comercializar produtos alimentícios por meio de *delivery*.

2.2 Objetivos Específicos

- Viabilizar à varejistas a venda *on-line* de seus produtos por meio de uma interface amigável;
- Oferecer um módulo de administração do sistema, que permita o cadastro e edição dos produtos disponíveis para a venda, assim como, configurações referentes a entrega;
- Permitir ao cliente realizar pedidos *online* e acompanhar sua situação;
- Possibilitar ao cliente cadastrar e escolher o endereço de entrega desejado;
- Redirecionar a listagem de produtos desse pedido para o *Whatsapp* da loja, permitindo que o contato com um atendente seja iniciado imediatamente, assim que o pedido seja finalizado no aplicativo;
- Oferecer opções para o cliente retirar na loja ou realizar *delivery*;
- Permitir o registro de pedidos no banco de dados, de forma que possibilite a consulta do que foi comprado, o horário para entrega e o valor total do pedido;
- Fornecer relatórios de controle aos varejistas, incluindo: formas de pagamento por pedido e faturamento diário.

1.3 JUSTIFICATIVA

A crescente demanda por aplicações *delivery* se mostra cada vez maior nos dias atuais. Não há dúvidas de que a pandemia da *Covid-19* acelerou essa demanda. Nota-se que com os *lockdowns* impostos aos comércios, o uso de aplicativos de *delivery* se mostra como alternativa promissora à alimentação fora do lar. A aplicação proposta nesse projeto, oferece uma alternativa para hortifrutigranjeiros e afins ao uso de *marketplaces* digitais. Nesse mercado, existe grande concorrência, altas tarifas sobre pedidos realizados e pouco contato com os clientes. Desta forma, um aplicativo próprio irá auxiliar tanto o empreendedor quanto seu cliente.

Sabendo que o aplicativo pode ser visto como uma extensão da loja física em meios virtuais, ressalta-se o aumento do reconhecimento da loja e expansão da marca que o mesmo pode motivar. Oferecendo novos canais de venda, tem-se a ampliação do faturamento da loja por meio de vendas *on-line* e maior alcance de público fora da área de atuação da loja física.

Para os clientes da loja, tem-se mais conveniência, praticidade e economia de tempo, dado que poderá escolher um horário de entrega do seu pedido na sua casa de acordo com sua própria disponibilidade. Nota-se também que, mesmo aqueles clientes que não utilizam lojas virtuais, valorizam as lojas com presença *on-line* (SCHIEFFELBEIN; MARTINS; FURIAN, 2011). Percebe-se que o aplicativo proposto estará beneficiando o estabelecimento com maior liberdade para apresentar seus produtos, por meio da codificação, visto que será possível personalizar a loja da forma que o proprietário desejar, algo que *marketplaces* ou aplicativos construídos com ferramentas *no-code*, por exemplo, não provêm.

Atualmente, ferramentas *no-code* são limitadas em suas funcionalidades, e demandam cobrança mensal para funcionar da forma que a empresa necessita. Sendo assim, neste trabalho foi desenvolvido um sistema *web* PWA que possibilite ao cliente de lojas de produtos hortifrutigranjeiros (ex. frutarias, quitandas) a realização de pedidos de produtos de forma *online*, com agilidade e segurança.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos. Este é o primeiro capítulo e apresenta as considerações iniciais com o contexto do sistema a ser desenvolvido, os seus objetivos e a justificativa. O Capítulo 2 apresenta a fundamentação teórica que compõe os conceitos necessários para o desenvolvimento deste trabalho e aponta a perspectiva de outros autores sobre os tópicos abordados. No Capítulo 3 estão as ferramentas e as tecnologias utilizadas na modelagem do sistema e que serão utilizadas na implementação subsequente do sistema. O Capítulo 4 Apresenta os resultados alcançados neste trabalho até este momento.

2 REFERENCIAL TEÓRICO

O referencial teórico contará com os principais tópicos abordados neste trabalho, sendo esses: *E-commerce*, *delivery* em hortifrutis e PWA.

2.1 E-COMMERCE

Por definição, comércio eletrônico (do inglês, *e-commerce*) é toda atividade de compra e venda realizada com o auxílio de recursos eletrônicos (NAKAMURA, 2001). O comércio eletrônico mostra-se como um meio para facilitar a compra e venda, fazendo uma verdadeira revolução no comércio de produtos e serviços, tornando o que era fácil, mais confortável, rápido e de baixo custo (FERREIRA DE ANDRADE; GONÇALVES DA SILVA, 2017). *E-commerce* permite que clientes comprem ou realizem transações em qualquer lugar, oferecendo a eles mais escolhas entre fornecedores e produtos, além da possibilidade de conduzir comparações de preços mais rapidamente. Permitindo que empresas interajam mais de perto com os clientes (MANZOOR, 2010).

A *Internet* também representa ameaças e oportunidades, por aumentar o poder de negociação dos compradores, deixando-os mais informados sobre produtos e preços, estando a um *click* do mouse de seus concorrentes (RAMOS et al., 2011). Por outro lado, tem-se o alcance de mais clientes, visto que esse formato de negócio permite que comerciantes atendam a uma quantidade grande de consumidores perto ou longe de sua localidade (MENDONÇA, 2016).

Atualmente, tem-se vivenciado a consolidação do varejo eletrônico como um importante canal de vendas (ALDAY; PINOCHET, 2002). Para a maioria dos consumidores, o *e-commerce* é mais um jeito de fazer compras, é um canal de vendas, porém, com mais comodidade, praticidade e rapidez (MENDES, 2013). Operando em uma economia altamente competitiva, empresas são levadas a adotar novos modelos de negócios. Nessas condições, o *e-commerce* se tornou prerrogativa de obtenção de sucesso no mercado para qualquer empresa (DONICI; IGNAT; MAHA, 2012).

Percebe-se que o segmento de *e-commerce* segue em expansão no Brasil. No mês de dezembro de 2020 registrou-se alta de 53,83% em relação ao mesmo período de 2019. O faturamento, considerando a mesma base comparativa, teve crescimento de 55,74% (*E-COMMERCE BRASIL*, 2021). Mendonça (2016) aponta que o *e-commerce* brasileiro tem crescido devido à quantidade de usuários cada vez maior e no aumento da confiança em

realizar as compras via Internet. O autor ainda afirma que o medo de efetuar compras pela de forma *on-line* tem diminuído devido à comodidade que a mesma proporciona.

Hoje em dia, a linha entre o *e-commerce* e o comércio tradicional está se tornando mais tênue à medida que mais empresas começam e continuam a integrar a Internet e as tecnologias de *e-commerce* em seus processos de negócios (GOEL, 2007). A constante expansão dos canais e ferramentas de consumo faz emergir um novo perfil de consumidor: o cliente que além de frequentar as lojas físicas, está em contato com os canais virtuais de vendas (SCHIEFFELBEIN; MARTINS; FURIAN, 2011). Esses clientes são importantes para *e-commerces* hortigranjeiros por manter o faturamento recorrente existente e adicionar a ele novas fatias de mercado.

2.2 DELIVERY EM HORTIGRANJEIROS

Segundo Mortimer et al. (2016), a experiência de comprar alimentos e mantimentos *on-line* é fundamentalmente diferente de outras formas de compras *on-line*, devido à perecibilidade e variabilidade do produto e à frequência das compras. Ao comprar mantimentos, os compradores *on-line* esperam a mesma qualidade do produto que obtêm quando eles próprios escolhem o produto à mão (SINGH, 2019).

Um dos obstáculos mais significativos para o crescimento do *e-commerce* em geral, e de hortifrutigranjeiros em particular, é a falta de uma infraestrutura logística adequada para entrega em domicílio (PUNAKIVI; SARANEN, 2001). No caso de produtos dos gêneros alimentícios, os custos de transporte são um importante fator para o sucesso do negócio. O período das janelas de tempo de entrega e localização da entrega também pode influenciar o custo geral de transporte (DA SILVA, 2018).

Pan et al. (2017), apresentam uma série de fatores que podem afetar a fidelização dos clientes para suas compras de supermercado, sendo eles:

1. Uma loja *on-line* bem projetada e fácil de usar;
2. Disponibilidade de uma gama de produtos;
3. Opções de entrega e logística adequadas;
4. Consistência entre todos os canais de vendas e mídia.

Reichheld e Teal (1996) indicam que clientes fiéis compram mais, com maior frequência e indicam a empresa para outros potenciais compradores, causando o crescimento de lucros de forma mais rápida quando comparado ao varejo físico. Portanto, afim de fidelizar o máximo de clientes, será desenvolvido um aplicativo web PWA para entregar uma boa

experiência para o usuário, aumentando o índice de satisfação e ampliando as chances de uma possível fidelização (GOBACKLOG, 2021).

2.3 PROGRESSIVE WEB APPS (PWA)

Qualquer *site* ou serviço que esteja hoje em funcionamento, terá grande parte de seus acessos oriundos de dispositivos móveis (celulares). E em muitas vezes é natural o pensamento de se desenvolver uma aplicação mobile. Atualmente é possível notar que a maioria dos acessos a *sites* ou serviços *on-line* ocorrem a partir de um celular, o que é acessível à maioria das pessoas. Essa realidade gera a necessidade de que esses *sites da web* sejam adaptáveis ao serem acessados por esses dispositivos. Nesse contexto, se tem os PWAs, *sites* que empregam tecnologia moderna para fornecer experiências semelhantes a aplicativos nativos na *web* (HEILMANN et al., 2018).

Sendo assim, os PWA usam recursos modernos da *Web* para oferecer uma experiência de usuário em ambiente móvel, semelhante a um aplicativo *mobile* (OSMANI, 2015). Eles oferecem recursos de envio de notificações aos usuários, atalhos na área de trabalho do dispositivo, disponível independentemente de conexão com a Internet, acesso a alguns recursos do sistema operacional como: câmera, galeria, geolocalização e agenda de contatos (SILVA NETO, 2020).

Esta nova tecnologia permite que uma aplicação possa estar disponível em qualquer dispositivo com acesso a um browser *Web*, sem a necessidade de desenvolver a aplicação nativa, especificamente para determinado dispositivo ou sistema operacional (FORTUNATO; BERNARDINO, 2018). Trindade e Affini (2018), definem uma aplicação PWA por um conjunto de conceitos e palavras-chave, incluindo:

- Conectividade progressiva (para qualquer usuário, independente do browser);
- Ser responsivo (elaborado para qualquer dispositivo *desktop* ou *mobile*);
- *App-like* semelhante a um aplicativo – (o usuário se sente em um aplicativo nativo);
- Atual (não há necessidade de fazer atualizações, como está tudo na *web*, quando abrir o aplicativo, a nova versão já será carregada);
- Descobrível - *SEO Friendly* – (os mecanismos de busca conseguem encontrar o conteúdo dos aplicativos);

- Reengajável (por meio de *push notifications*¹, o usuário pode ser constantemente engajável); instalável (podem ser adicionados à tela inicial do dispositivo móvel);
- Ser acessível (mais fácil de compartilhar conteúdo ao enviar o link de acesso para alguém).

Chris Heilmann et al. (2018) apresentam três características que PWAs devem atender:

1. Rápido - o primeiro contato com uma interface PWA é uma experiência imediata, responsiva e agradável.
2. Instalável - um PWA é instalado como qualquer outro aplicativo nativo, mostrando na sua área de trabalho ou tela inicial, em vez do que exigir um navegador.
3. Engajador - o PWA segue as melhores práticas para *UX* e *UI* de alta qualidade e funciona independentemente do estado da rede ou da capacidade do dispositivo.

Os PWA começam como *sites* simples, mas à medida que o usuário se envolve com eles, transformam um *site* em algo muito parecido com um aplicativo nativo tradicional. Este novo modelo progressivo substitui a natureza binária instalada/não instalada de aplicativos nativos (ATER, 2017). Essa instalação instantânea, a experiência de navegação *offline* e os recursos de notificação para o usuário, atraem proprietários de *sites* e os motiva a implementar seus *sites* com PWAs (LEE; KIM; PARK et al., 2018). São exemplos de aplicativos PWA:

2.3.1 Trivago

Trivago é uma plataforma de busca de hotéis, com a expansão da plataforma para diversos países, o investimento em uma aplicação PWA foi a maneira que a Trivago encontrou para tornar a jornada dos seus usuários mais agradável e eficiente, independente do tipo de conexão que ele está usando. Usuários podem navegar com a plataforma *offline* (TRIVAGO, 2021).

2.3.2 Twitter

A grande maioria (80%) dos usuários do Twitter acessam usando plataformas móveis, portanto, eles precisam de um aplicativo leve que possa fornecer acesso mais rápido e eficiente aos dados de seu aplicativo (SOUZA, 2021). O Twitter foi um dos pioneiros do PWA

¹ *Push notification* é uma notificação que o usuário recebe em um aplicativo de *smartphone*, tablet ou em um navegador sem requisitá-la.

em 2017. A introdução do Twitter Lite, a versão PWA do twitter aumentou as visualizações de páginas por sessão em 65%, os tweets enviados em 75% e as taxas de retorno de tela foram reduzidas em 20% (LOVE, 2021) (TWITTER, 2021).

3.2.3 Spotify

O Spotify, um serviço de *streaming* de música digital com milhões de músicas e *podcasts*, é outro exemplo de PWA para *desktop*. O Spotify pode ser acessado diretamente pelo navegador permitindo adicionar um atalho na tela inicial do aparelho, sem a necessidade de baixar o aplicativo (SPOTIFY, 2021).

2.4 SISTEMAS HORTIFRUTIGRANJEIROS EXISTENTES

A seguir serão apresentados alguns exemplos de sistemas de *e-commerces* para produtos hortigranjeiros e suas características, os quais se encontram em funcionamento.

2.4.1 Hortifruti *Delivery*

Uma franquia com 43 lojas, contendo um *site* principal e um *site* para *delivery*. O *site* conta com promoções de temporada que oferecem cupons de desconto. Sua estrutura é responsiva, apresentando produtos separados por categorias e pedido com valor mínimo de 50 reais para entrega sem frete, atendendo somente bairros específicos. Possui uma funcionalidade de informar se o bairro é atendido pela loja assim que o cliente informa seu CEP (HORTIFRUTI, 2021).

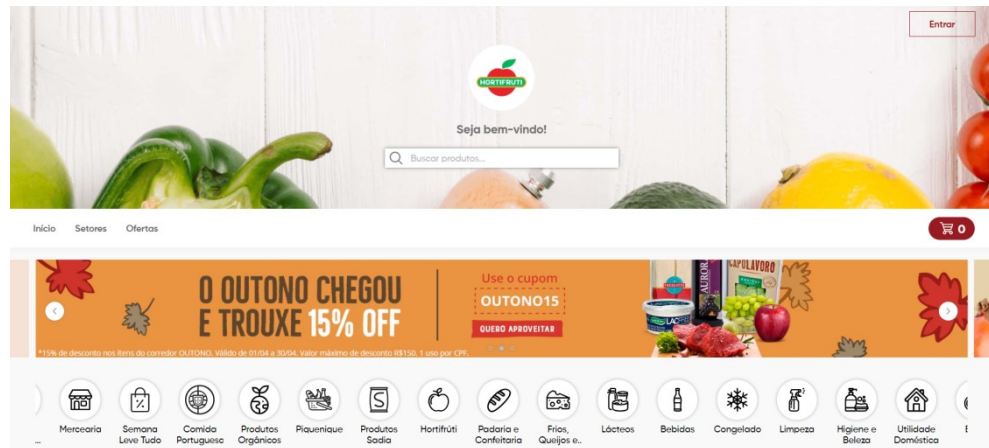


Figura 1 – Tela inicial do e-commerce delivery hortifruti

Fonte: delivery.hortifruti.com.br.

2.4.2 Frutello Horifruti

Desenvolvido por uma ferramenta *no-code*, esta aplicação PWA conta com uma aba de produtos, carrinho e informações da empresa. Possui produtos separados por categorias e promoções especiais, conta também com uma barra de filtragem que passa por todos os produtos disponíveis. Ao finalizar a compra, a aplicação encaminha o usuário para o *whatsapp* da empresa para confirmação do pedido (FRUTELLO, 2021).

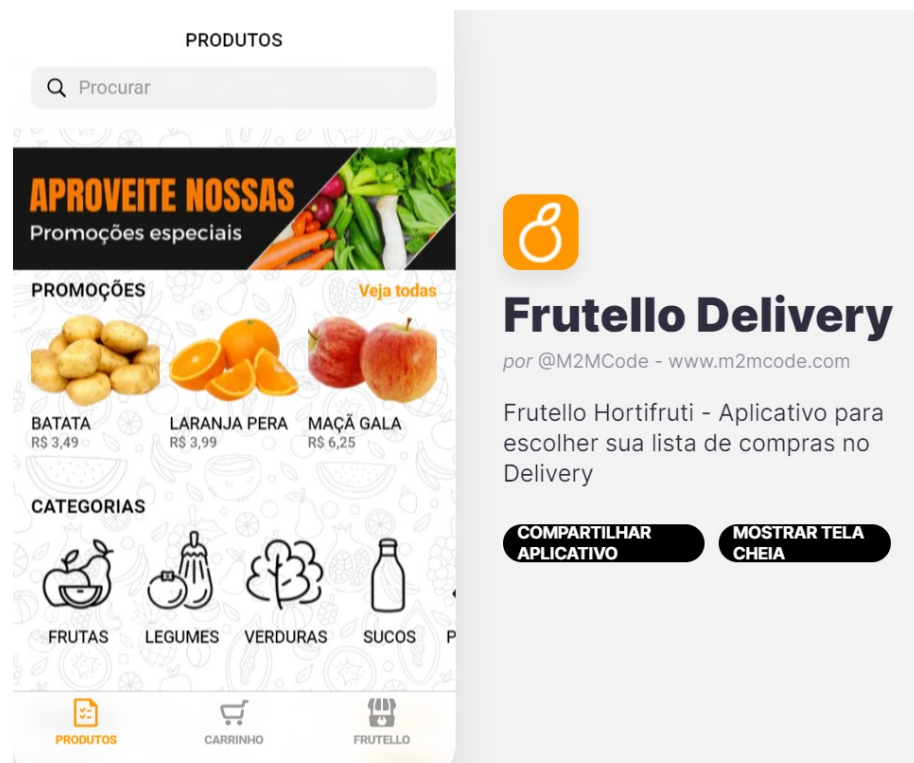


Figura 2 -Tela inicial da aplicação frutello delivery

Fonte: frutellodelivery.glideapp.

2.4.3 Orgânicos *In box*

A *Orgânicos in Box* é uma plataforma digital para *delivery* de produtos 100% orgânicos, que conecta pequenos produtores rurais a consumidores urbanos. Conta com cestas prontas e frete grátis (ORGÂNICOS IN BOX, 2021).



Figura 3 – Tela inicial do e-commerce Organicos In Box

Fonte: organicosinbox.com.br

2.4.4 *iFood*

O *iFood marketplace* apresenta a proposta de ser um *marketplace* digital que concentra em um único lugar, *website* ou *app*, uma grande variedade de restaurantes e seus cardápios para clientes em diferentes localidades, também oferecendo produtos hortigranjeiros. O *iFood* conta com a participação da Movable, líder global em *marketplaces* móveis e da Just Eat (IFOOD, 2021).

ifood Shop

Rio de+Janeiro Seja nosso fornecedor Entre ou cadastre-se

Olá, o que você procura?

Carrinho

Ofertas da Semana

Mercearia Bebidas Queijos e Frios Açougue e Peixaria Embalagens Mais Categorias

Hortifruti (190)

Hortifruti x

Faixa de preço de R\$ até R\$

Estado Rio de Janeiro (190)

Categorias Hortifruti (190)

Loja Irmãos Bena... (188) Playvender (2)

Marcas

Frutas, legumes e Verduras

Direto do CEAGESP na porta do seu restaurante. Após a confirmação de pagamento, entrega em até 24h. Horário de corte: 18h00






 <p>Abacate Aa Benassi Caixa com 1.0 Kg 1 KG</p> <p>R\$3,20</p> <p>Adicionar</p>	 <p>Abacaxi Perola Extra G Abacaxi Perola Extra ... 1 KG</p> <p>R\$3,20</p> <p>Adicionar</p>	 <p>Abobora Japonesa Grauda / Cabotian... 1 KG</p> <p>R\$3,00</p> <p>Adicionar</p>	 <p>Abobora Sergipana Abobora Sergipana... 1 KG</p> <p>R\$3,50</p> <p>Adicionar</p>	 <p>Abobrinha Brasileira Aa Benassi Caixa com 1.0... 1 KG</p> <p>R\$2,60</p> <p>Adicionar</p>
---	---	---	---	--

Figura 4 – Página de hortifruti no marketplace Ifood

Fonte: Ifood

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a realização deste trabalho. Os materiais estão relacionados às tecnologias e ferramentas utilizadas e o método apresenta a sequência das principais atividades realizadas.

3.1 MATERIAIS

A aplicação foi desenvolvida utilizando o Laravel, um *framework* de aplicação web na linguagem de programação PHP e o *framework* CSS Bootstrap 4 para o desenvolvimento responsivo. No quadro 1 estão listados os materiais que serão utilizados para o desenvolvimento do trabalho e implementação da aplicação.

Ferramenta / Tecnologia	Versão	Finalidade
Visual Paradigm	16.2	Modelagem do sistema
MYSQL Workbench	8.0.11	Modelagem do banco de dados
MySQL	8.0.22	Banco de dados
PHP	7.4.16	Linguagem de Programação
Javascript	ES12	Linguagem de Programação
Jquery	3.3	Biblioteca de funções Javascript
HTML	5	Linguagem de Estruturação e Marcação de Conteúdo
CSS	3	Linguagem de Estilização
Visual Studio Code	1.62.1	IDE de desenvolvimento
Laravel	6.20.34	<i>Framework</i> back-end
Artisan	6	Interface de linha de comando usada no Laravel
Composer	v2.1.12	Ferramenta para gerenciamento de dependências em PHP
Blade Engine	6	Engine de templates do <i>framework</i> Laravel
Font Awesome	5.15.4	Kit de ferramentas de ícones
Glideapps	V6	Prototipação do aplicativo
Google Lighthouse	2.8	Ferramenta automatizada para auditar os aplicativos da web

Quadro 1 - Lista de ferramentas e tecnologias

3.2 MÉTODO

Para a implementação do método foi utilizado o Processo Unificado, o qual foi definido pelos autores da Linguagem de Modelagem Unificada - UML, a fim de disponibilizarem um modelo de processo que permitisse aplicar a UML em todas as suas etapas. Sendo assim, Booch, Rumbaugh e Jacobson (2006), propuseram um modelo iterativo e incremental no qual todas as etapas de desenvolvimento de software padrão (levantamento de requisitos, análise, projeto, implementação, testes e implantação) são repetidas, gerando a cada iteração um sub-projeto e um produto funcional parcialmente concluído. Desta forma, o ciclo de desenvolvimento contará com as seguintes etapas.

1. Prototipação do sistema

A prototipação do sistema foi realizada com o auxílio da ferramenta *no-code* Glide com o aplicativo frutello-delivery.glideapp.io. Inicialmente desenvolvida para implementação em um estabelecimento comercial. O sistema conta com controle de produtos, adição de produtos no carrinho e efetuação de pedidos, principais requisitos necessários para o funcionamento de um *delivery*. Logo, o produto final desenvolvido pelo presente trabalho se baseará nesta aplicação efetuando melhorias pontuais a adicionando funcionalidades extras conforme identificado na etapa de análise do sistema.

2. Análise do sistema

Nessa fase, o sistema começou a ser concebido a partir da análise de requisitos, suas funcionalidades e requisitos funcionais e não-funcionais foram descritos, a fim de apresentar as necessidades dos usuários da futura aplicação. Em seguida, foi realizada a modelagem dos principais processos de negócio por meio de diagrama de atividades, objetivando clarear o entendimento das funcionalidades que o aplicativo deverá apresentar. Foram também elaborados diagramas de caso de uso, assim como a expansão dos principais processos do sistema. Por fim, foi criado o modelo conceitual, o que apresenta o modelo de classes preliminar para o sistema. Foi utilizada a ferramenta Visual Paradigm para a elaboração dos artefatos dessa fase.

3. Projeto do sistema

A fase de projeto também se utilizou da ferramenta Visual Paradigm. Essa fase é composta por diagramas de sequência, de forma a identificar os métodos para as futuras classes do sistema. Em seguida, foi criado o diagrama de classes de projeto, bem como o diagrama entidade-relacionamento (DER), utilizando a ferramenta Mysql Workbench.

4. Codificação do sistema

A codificação do sistema contou com a utilização da IDE Visual Studio Code, as linguagens de programação PHP, HTML, Javascript e CSS e demais tecnologias, conforme descritas no Quadro 1. O *framework* de código aberto Laravel foi usado para tornar o desenvolvimento da aplicação mais ágil por meio de recursos integrados como: sistema de autenticação, mapeamento objeto-relacional e interface de linha de comando (Artisan) que vem com comandos pré-construídos. Tendo como base as etapas de prototipação, análise e projeto do sistema para as funcionalidades e *layout* de telas, seguindo seus modelos e requisitos identificados. Ademais, para que a aplicação seja considerada um PWA e se tenha controle da experiência do usuário, foi preciso manter os padrões definidos pelo Google (HEILMANN et al., 2018).

5. Fase de testes

Os testes foram realizados a cada iteração de acordo com as especificações dos casos de uso para o sistema. Além disso, houve a participação de futuros usuários do sistema, opinando sobre cada sub-produto gerado e sugerindo melhorias a serem adaptadas. Com isso foi possível incrementar tanto o documento de especificação de requisitos, quanto as funcionalidades implementadas e sua interface.

4 ANÁLISE E PROJETO DO SISTEMA

Este capítulo apresenta o resultado da modelagem e o desenvolvimento do sistema *web* apresentado no presente trabalho.

4.1 ESCOPO DO SISTEMA

4.1.1 Definição do Sistema

O sistema é uma aplicação *e-commerce* para realização de pedidos de produtos hortifrutigranjeiros via *Web*. A aplicação contará com gestão de produtos em estoque, carrinho e pedidos, além de controlar o *delivery* dos produtos pedidos. Sendo assim, o principal objetivo do sistema é trazer uma loja física já existente nesse setor para a Internet, como uma ferramenta para a venda de produtos hortifrutigranjeiros de forma *online*. Além da loja física ter mais visibilidade com mais um canal de venda sem depender de *marketplaces*.

4.1.2 Principais Processos do Sistema

A aplicação de *delivery* oferece as seguintes funcionalidades:

1. Cadastrar usuários, clientes e funcionários;
2. Autenticar usuários;
3. Cadastrar e editar produtos a serem disponibilizados na loja virtual;
4. Controlar carrinho de compras, permitindo adicionar novos produtos e removendo quando necessário, com opção de alterar a quantidade;
5. Cadastrar endereços de entrega;
6. Registrar pedidos, com opção para escolha do horário de entrega, forma de pagamento e cálculo do valor da compra com e sem entrega.
7. Cadastrar e alterar dados da empresa;
8. Redirecionar mensagem com pedido para o *whatsapp* do estabelecimento.
9. Geração de relatórios básicos de controle para pagamentos recebidos.

4.2 MODELAGEM DO SISTEMA

Esta seção apresenta os requisitos funcionais e não funcionais identificados no sistema, assim como diagrama de caso de uso, diagramas sequenciais, diagrama de atividades, diagrama de classes e diagrama entidade relacionamento.

F1 Cadastrar produtos		Oculto ()		
Descrição: O sistema possibilitará o cadastro dos produtos a serem comercializados. De forma que sejam registrados seus detalhes como: nome, categoria, preço, descrição, imagem do mesmo e forma de venda (vendido por peso em quilograma ou unidade)				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF1.1] Apresentar produtos disponíveis	O sistema somente deve apresentar produtos que constam como “disponível” no status para o cliente.	Regra de negócio/ Interface	()	(x)
[NF1.2] Controlar nome do produto	Cada produto deve conter uma identificação e descrição únicos.	Regra de negócio	()	(x)
[NF1.3] Separar por categoria	Os produtos serão cadastrados por categoria.	Interface	(x)	(x)
[NF1.4] Cadastrar imagem	A imagem do produto deve ser cadastrada sendo armazenada como varchar no banco de dados.	Interface	()	(x)

Quadro 2 – Requisito Cadastrar Produto

F2 Listar produtos		Oculto ()		
Descrição: O sistema listará os produtos previamente cadastrados, permitindo ao cliente selecioná-los.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF2.1] Listar por disponibilidade	Os produtos somente serão listados se estiverem com o valor “Disponível” no campo <i>status</i> .	Interface	()	(x)
[NF2.2] Organizar por ordem alfabética	Na listagem, os produtos serão organizados por ordem alfabética.	Interface	(x)	(x)

Quadro 3 – Requisito Listar Produtos

F3 Controlar carrinho de compras		Oculto ()		
Descrição: O sistema permite adicionar e remover produtos e sua quantidade desejada no carrinho de compra.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF3.1] Notificar cliente	Ao clicar em adicionar produto no carrinho, o	Interface	(x)	(x)

	sistema deve mostrar uma notificação de erro ou sucesso, conforme a disponibilidade da quantidade desejada.			
--	---	--	--	--

Quadro 4 – Requisito Controlar carrinho de compras

F4 Informar endereços de entrega		Oculto ()		
Descrição: Deve ser permitido cadastrar novos endereços de entrega após o cadastro do usuário.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF4.1] Apresentar taxa de entrega	O sistema deve informar se o endereço selecionado terá taxa de entrega ou não.	Regra de negócio/ Interface	(x)	(x)

Quadro 5 – Requisito Informar endereços de entrega

F5 Escolher horário de entrega		Oculto ()		
Descrição: Durante a realização do pedido, o sistema disponibilizará uma lista de horários que o cliente poderá receber sua entrega, levando em consideração o horário de funcionamento da loja e a quantidade de entregas nesse período.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF5.1] Apresentar horários disponíveis	O sistema deve limitar os horários selecionáveis de acordo com a quantidade de entregas permitidas. Apresentando os horários disponíveis para entrega da mercadoria para que o usuário escolha um destes horários permitidos.	Regra de negócio/ Interface	(x)	(x)

Quadro 6 – Requisito Escolher horário de entrega

F6 Calcular valor aproximado da compra		Oculto (x)		
Descrição: O valor total aproximado do pedido deve ser calculado baseado na quantidade, preço dos produtos e taxa de entrega (a depender do endereço informado).				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF6.1] Exibir mensagem de aviso	O sistema deve avisar o cliente que o valor apresentado é aproximado. O verdadeiro valor dependerá da disponibilidade dos produtos no estoque, bem como de pequenas variações em seu peso. O valor apresentado ao cliente nunca deverá ser ultrapassado.	Interface	(x)	()
[NF6.1] Não cobrar taxa de entrega	O sistema não cobrará taxa de entrega quando o pedido possuir o endereço próximo ao estabelecimento. A	Regra de negócio/ Interface	()	(x)

	distância limite para entrega gratuita, assim como o valor da taxa e bairros a serem cobrados serão definidos pelo administrador nas configurações da empresa.			
--	--	--	--	--

Quadro 7 – Requisito Calcular valor aproximado da compra

F7 Registrar pedido		Oculto (x)		
Descrição: O requisito estará disponível após os requisitos RF03, RF04 e RF05 serem concluídos. O sistema deverá armazenar os produtos dos pedidos, suas respectivas quantidades, nome, endereço para entrega, horário de entrega, valor total da compra e forma de pagamento (dinheiro, crédito, débito, vale refeição, PIX).				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF7.1] Armazenar pedido	Caso o usuário esteja logado e possua o perfil de usuário como cliente, o sistema deve vincular o pedido salvo ao seu registro.	Regra de negócio/ Interface	(x)	(x)
[NF7.2] Informar endereço de entrega	Caso o cliente não possua cadastro anterior, o sistema deve receber o endereço de entrega durante a realização do pedido, para o qual o sistema deverá retornar se haverá taxa de entrega ou não.	Regra de negócio/ Interface	()	(x)
[NF7.3] Enviar pedido via whatsapp	Ao finalizar sua compra e registrar seu pedido, o sistema deve encaminhar o pedido como mensagem para o <i>whatsapp</i> da loja.	Regra de negócio	(x)	()

Quadro 8 – Requisito Registrar pedido

F8 Listar pedidos		Oculto ()		
Descrição: O requisito estará disponível após o pedido ser efetuado pelo cliente. O sistema deverá listar todos os pedidos em uma tela separado permitindo a edição somente do status.				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF7.1] Listar pedidos para clientes	Caso o perfil seja do tipo cliente, apenas poderá visualizar os pedidos associados com seu cadastro.	Regra de negócio/ Interface	(x)	(x)
[NF7.2] Listar pedidos para o administrador	Todos os pedidos serão apresentados, caso o perfil seja do tipo administrador	Regra de negócio/ Interface	()	(x)
[NF7.3] Alterar status do pedido apresentado	É permitido a edição do status de cada item da lista. Caso o perfil seja vinculado com a loja. Os status podem ser “Aguardando, Enviado para separação, Enviado para entrega ou Entregue.	Regra de negócio/ Interface	(x)	(x)

Quadro 9 – Requisito Listar pedidos

F9 Cadastrar usuários		Oculto ()		
Descrição: O sistema deve permitir o cadastro de usuários com nome, e-mail, senha e perfil de usuário (cliente ou funcionário), endereço de entrega, formas de pagamento (ex. dinheiro, cartão, pix).				
Requisitos Não-Funcionais				
Nome	Restrição	Categoria	Desejável	Permanente
[NF9.1] Cadastrar perfil de usuário	O administrador poderá cadastrar perfis de usuários.	Regra de negócio/ Interface	()	(x)
[NF9.2] Salvar endereço do cliente	Usuários clientes poderão cadastrar possíveis endereços de entrega, para não precisar preencher esse campo novamente, em caso de novos pedidos.	Regra de negócio/ Interface	(x)	(x)
[NF9.3] Permissão do administrador	Para finalizar o cadastro de usuários com o perfil de funcionário, o administrador deverá aceitar a requisição.	Regra de negócio/ Interface	()	(x)

Quadro 10 – Requisito Cadastrar usuários

F10 Cadastrar dados da empresa	Oculto ()
Descrição: Permitir ao usuário com o perfil de administrador alterar os dados da empresa, esses dados contém: nome, endereço completo, link <i>whatsapp</i> , facebook, instagram, telefone fixo, celular, imagem da empresa, bairros atendidos pela empresa, horário de atendimento, horários de entrega e quantidade máxima de entregas possíveis dentro de cada horário, distância máxima em que não será cobrada taxa de entrega. Deverá ser permitido alterar todos esses dados.	

Quadro 11 – Requisito Cadastrar dados da empresa

O diagrama de caso de uso apresentado na Figura 5, contém as principais funcionalidades do sistema de *delivery* realizadas pelos atores, sendo os perfis cliente, funcionário e administrador, todos usuários do sistema. Todos usuários podem se cadastrar e realizar login, clientes poderão adicionar produtos no carrinho, efetuar e consultar pedidos realizados, perfis do tipo funcionário poderão cadastrar e alterar produtos do sistema e o perfil administrador além de possuir todas as permissões do perfil funcionário poderá gerenciar os usuários do sistema, alterar dados da empresa e registrar pagamentos.

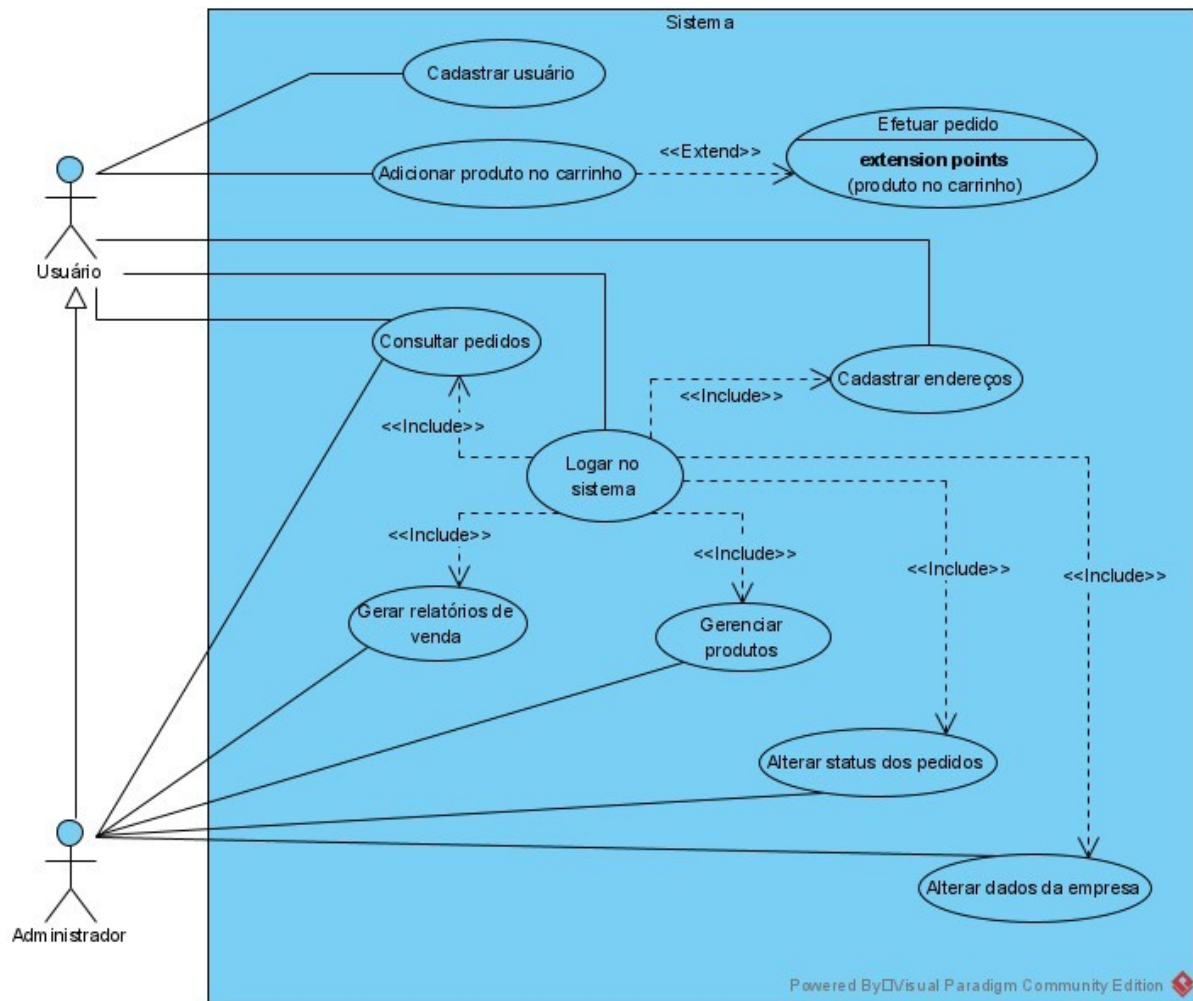


Figura 5 – Diagrama de caso de uso

O Quadro 12 apresenta o caso de uso cadastrar cliente:

Caso de uso: Cadastrar Usuário
Atores: Usuários e Administrador
Precondições: nome de usuario, senha, email, celular e imagem.
Pós-condições: dados do usuário inseridos no banco de dados, usuário pode realizar login no sistema
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicial quando: <ol style="list-style-type: none"> 1. [IN] O ator seleciona a opção cadastrar-se na tela inicial 2. [OUT] O sistema apresenta a tela de cadastro 3. [IN] O usuário informa os dados desejados 4. [OUT] O sistema verifica os dados informados

Quadro 12 – UC1 - Cadastrar Usuário.

O Quadro 13 apresenta o caso de uso logar no sistema:

Caso de uso: Logar no sistema
Atores: Usuários e Administrador
Precondições: Usuário cadastrado, digitar usuário e senha corretos
Pós-condições: Ator logado no sistema

Sequência típica de eventos (Fluxo Principal):

Esse caso de uso inicial quando:

1. [IN] O ator informa o usuário e senha.
2. [OUT] O sistema verifica se usuário e senha estão cadastrados.
3. [OUT] Ator tem acesso a tela de usuário

Tratamento de Exceções e Variantes:

Exceção 1a: Nome de usuário ou senha não encontrados.

- 1.1 O sistema informa ao ator que os dados não conferem.
- 1.2 O sistema disponibiliza opção de alteração de senha ao ator
2. 1.3 Retorna a tela de login.

Quadro 13 – Caso de uso logar no sistema

O Quadro 14 apresenta o caso de uso adicionar produto no carrinho.

Caso de uso: Adicionar produto no carrinho de compras

Atores: Usuários

Precondições: produto com status “Disponível”.

Pós-condições: Produto adicionado ao carrinho de compras.

Sequência típica de eventos (Fluxo Principal):

Esse caso de uso inicia quando:

1. [OUT] O sistema verifica o status do produto.
 - Variante 1.1: Produto indisponível
 - 1.1.1 [OUT] O sistema não lista o produto
 - Variante 1.2: Produto disponível
 - 1.1.2 [OUT] O sistema apresenta o produto para seleção
 - 1.1.3 [IN] O cliente seleciona o produto que deseja adicionar ao carrinho
 - 1.1.4 [OUT] O sistema apresenta uma nova página contendo os detalhes dos produtos e o campo “Quantidade”
 - 1.1.5 [IN] O cliente insere a quantidade desejada e clica em “Adicionar ao carrinho”
 - 1.1.6 [OUT] O sistema calcula o valor do pedido, adiciona o produto ao carrinho e solicita ao cliente se deseja finalizar a compra ou “continuar comprando”.

Quadro 14 – Caso de uso Adicionar produto no carrinho de compras

O Quadro 15 apresenta o caso de uso efetuar pedido.

Caso de uso: Efetuar pedido.

Atores: Usuários

Precondições: Usuário logado, Produtos adicionados no carrinho

Pós-condições: Pedido registrado no banco de dados e enviado para o *whatsapp* da loja

Sequência típica de eventos (Fluxo Principal):

Esse caso de uso inicia quando:

1. [IN] O cliente seleciona o botão “Finalizar Pedido”.
2. [OUT] O sistema apresenta um campo no formato de caixa de seleção “Forma de pagamento”

Variante 1.1: Cliente seleciona a opção “Dinheiro”

[OUT] O sistema apresenta um campo “Troco”

<p>[IN] O cliente informa o troco desejado Variante 1.2 Cliente seleciona “Cartão” [OUT] O sistema adiciona observação no pedido “Levar máquina”</p> <p>3. [OUT] O sistema apresenta o campo “Forma de retirada”</p> <p>Variante 3.1: Cliente seleciona a opção “Retirar na loja” [OUT] O sistema apresenta um campo “Nome” [IN] O cliente informa o nome [OUT] O sistema apresenta um campo para o cliente selecionar o horário de retirada [IN] O cliente seleciona o horário desejado</p> <p>Variante 3.2: Cliente seleciona a opção “<i>Delivery</i>” [OUT] O sistema apresenta um campo “Nome” [IN] O cliente informa o nome [OUT] O sistema apresenta os endereços cadastrados pelo cliente [IN] O cliente seleciona o endereço de entrega desejado [OUT] O sistema verifica se bairro selecionado possui taxa de entrega</p> <p>Variante 3.2.1: Bairro informado possui taxa de entrega [OUT] O valor da taxa é adicionado ao valor total do pedido</p> <p>Variante 3.2.2: Bairro informado não possui taxa de entrega [OUT] O valor total do pedido é apresentado para o cliente</p> <p>[OUT] O sistema apresenta o botão “Enviar para <i>whatsapp</i>”</p> <p>4. [IN] O cliente seleciona o botão 5. [OUT] O carrinho é registrado como pedido 6. [OUT] O pedido é encaminhado como mensagem de <i>whatsapp</i> para o estabelecimento</p>
<p>Tratamento de Exceções e Variantes: Exceção 1ª: Ator informa valor de troco menor que o valor total</p> <ol style="list-style-type: none"> 1. Sistema bloqueia a finalização do pedido e apresenta a mensagem “O troco deve ser igual ou maior que o valor total”

Quadro 15 – Caso de uso Efetuar pedido

O Quadro 16 apresenta o caso de uso Consultar pedidos

Caso de uso: Consultar pedidos
Atores: Usuários
Precondições: Pedido efetuado pelo cliente
Pós-condições: Histórico de pedidos apresentados para um usuário
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicial quando: <ol style="list-style-type: none"> 1. [OUT] O sistema valida o perfil do usuário Variante 1.1: Perfil do usuário não é do tipo “Administrador”

<p>1.1.1. [OUT] O sistema apresenta o histórico de compras do cliente através de uma lista de pedidos associados ao seu cadastro</p> <p>1.1.2. [IN] O cliente clica em ver mais detalhes</p> <p>1.1.3. [OUT] O sistema apresenta os detalhes do pedido para o cliente</p> <p>Variante 1.2: Perfil do usuário é do tipo “Administrador”</p> <p>1.1.1. [OUT] O sistema apresenta todos os pedidos para o usuário</p> <p>1.1.2. [IN] O usuário seleciona o pedido</p> <p>1.1.3. [OUT] O sistema apresenta os detalhes do pedido selecionado</p>
--

Quadro 16 – Caso de uso Consultar pedidos

O Quadro 17 apresenta o caso de uso Gerenciar produtos

Caso de uso: Gerenciar produtos
Atores: Administrador
Precondições: Nome do produto, valor, imagem, unidade de medida, descrição
Pós-condições: produto cadastrado no banco de dados do sistema, produto pode ser editado por um usuário autorizado
<p>Sequência típica de eventos (Fluxo Principal):</p> <p>Esse caso de uso inicial quando:</p> <ol style="list-style-type: none"> 1. [IN] O administrador informa os dados do produto. 2. [OUT] O sistema valida os dados inseridos 3. [IN] Produto inserido no sistema 4. [OUT] Sistema mostra mensagem de sucesso para o usuário
<p>Tratamento de Exceções e Variantes:</p> <p>Exceção 1ª: Campo em branco</p> <ol style="list-style-type: none"> 1. Apresenta mensagem de erro. 2. Destaca em vermelho os campos necessários

Quadro 17 – Caso de uso Gerenciar produtos

O Quadro 18 apresenta o caso de uso alterar informações da empresa

Caso de uso: Alterar dados da empresa
Atores: Administrador
Precondições: Usuário com perfil Administrador
Pós-condições: informações alteradas no banco de dados
<p>Sequência típica de eventos (Fluxo Principal):</p> <p>Esse caso de uso inicial quando:</p> <ol style="list-style-type: none"> 1. [IN] Administrador seleciona botão de alterar dados da empresa 2. [OUT] Sistema apresenta tela com campo para inserir senha do administrador 3. [IN] Administrador informa sua senha 4. [OUT] Sistema apresenta informações da empresa 5. [IN] Administrador altera dados da empresa 6. [OUT] O sistema valida os dados inseridos 7. [IN] Sistema atualiza informações no banco de dados 8. [OUT] Sistema mostra mensagem informações da empresa alteradas com sucesso
<p>Tratamento de Exceções e Variantes:</p> <p>Exceção 1ª: Campo em branco</p> <ol style="list-style-type: none"> 1. Apresenta mensagem de erro. 2. Destaca em vermelho os campos necessários

Quadro 18 – Caso de uso Alterar dados da empresa

O Quadro 19 apresenta o caso de uso Alterar status dos pedidos.

Caso de uso: Alterar status dos pedidos
Atores: Administrador
Precondições: Usuário com perfil Administrador
Pós-condições: Status do pedido alterado
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicial quando: <ol style="list-style-type: none"> 9. [IN] Administrador seleciona pedido 10. [OUT] O sistema mostra os detalhes do pedido 11. [IN] O administrador seleciona a opção com o status atual do pedido “Pedido em separação”, “Pedido à caminho”, ou “Pedido entregue” 12. [IN] O sistema altera o status do pedido para o status selecionado 13. [OUT] Sistema mostra mensagem status do pedido alterado com sucesso

Quadro 19 – Caso de uso Alterar status dos pedidos

O Quadro 20 apresenta o caso de uso Cadastrar endereços

Caso de uso: Cadastrar endereços
Atores: Usuários
Precondições: Cliente com usuário cadastrado no sistema
Pós-condições: Endereço adicionado ao seu usuário
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicial quando: <ol style="list-style-type: none"> 1. [IN] Usuário seleciona perfil 2. [OUT] O sistema mostra a tela de perfil 3. [IN] O usuário seleciona a opção “Endereços” 4. [OUT] O sistema apresenta o modal para cadastro de novo endereço 5. [IN] O usuário informa os dados do endereço 6. [OUT] Sistema mostra mensagem status de endereço adicionado com sucesso

Quadro 20 – Caso de uso Cadastrar endereços

O Quadro 21 apresenta o caso de uso Gerar relatórios

Caso de uso: Gerar relatórios de vendas
Atores: Administrador
Precondições: Administrador logado no sistema, acessando o painel do administrador
Pós-condições: Relatório de vendas gerado
Sequência típica de eventos (Fluxo Principal): Esse caso de uso inicial quando: <ol style="list-style-type: none"> 1. [IN] Usuário seleciona painel do administrador 2. [OUT] O sistema apresenta o painel 3. [IN] O usuário seleciona a opção “Relatório de vendas” 4. [OUT] O sistema apresenta os filtros para o usuário 5. [IN] O usuário filtra o relatório de acordo com sua necessidade 6. [OUT] Sistema apresenta o relatório de pedidos filtrado para o usuário

Quadro 21 – Caso de uso Gerar relatórios de venda

A Figura 6 apresenta o diagrama sequencial de adicionar no carrinho.

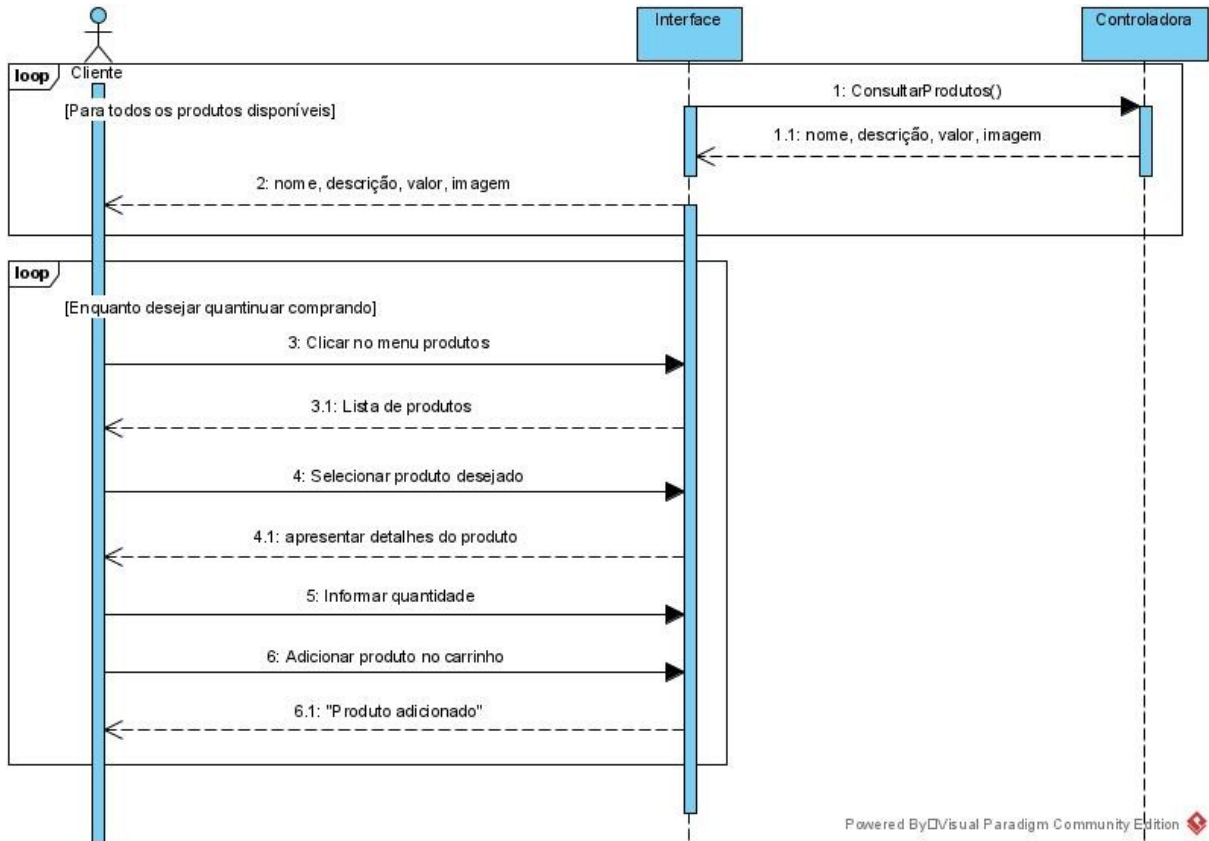


Figura 6 – Diagrama de seqüência para adicionar no carrinho

A Figura 7 apresenta o diagrama sequencial de adicionar no carrinho.

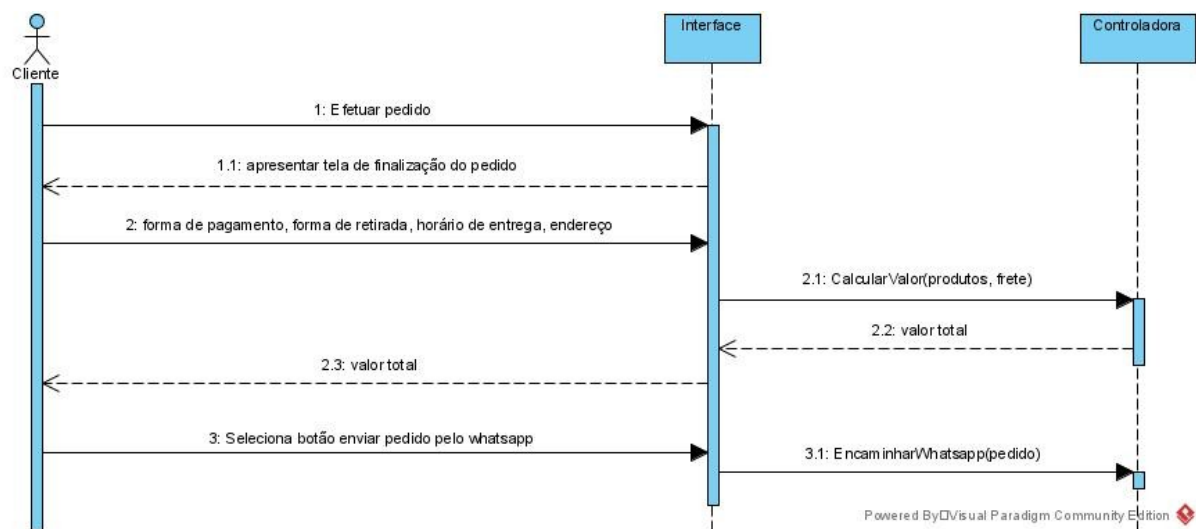


Figura 7 – Diagrama de seqüência para efetuar pedido

A Figura 8 apresenta o diagrama sequencial que representa o usuário administrador consultando pedidos.

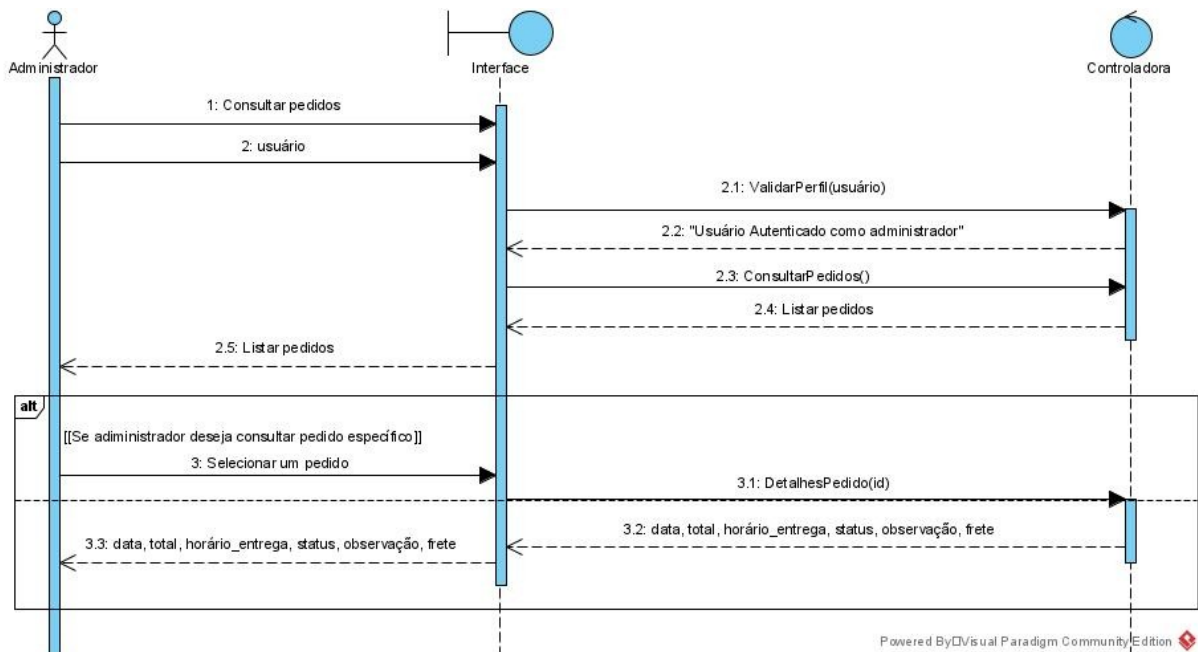


Figura 8 – Diagrama de seqüência para consultar pedidos

A Figura 9 apresenta o diagrama sequencial que representa o usuário administrador alterando informações da empresa.

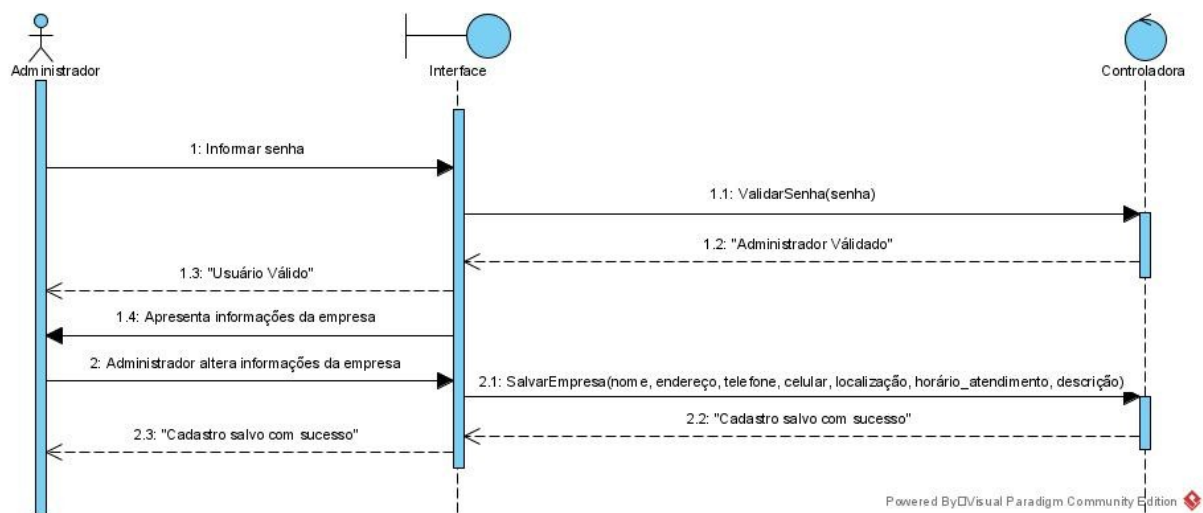


Figura 9 – Diagrama de seqüência para alterar informações da empresa

A Figura 10 apresenta o diagrama de atividades para o processo de efetuar pedido, englobando as funcionalidades de listar produtos, inserir no carrinho, login do usuário, enviar pedido por *whatsapp* e Lançar pagamento.

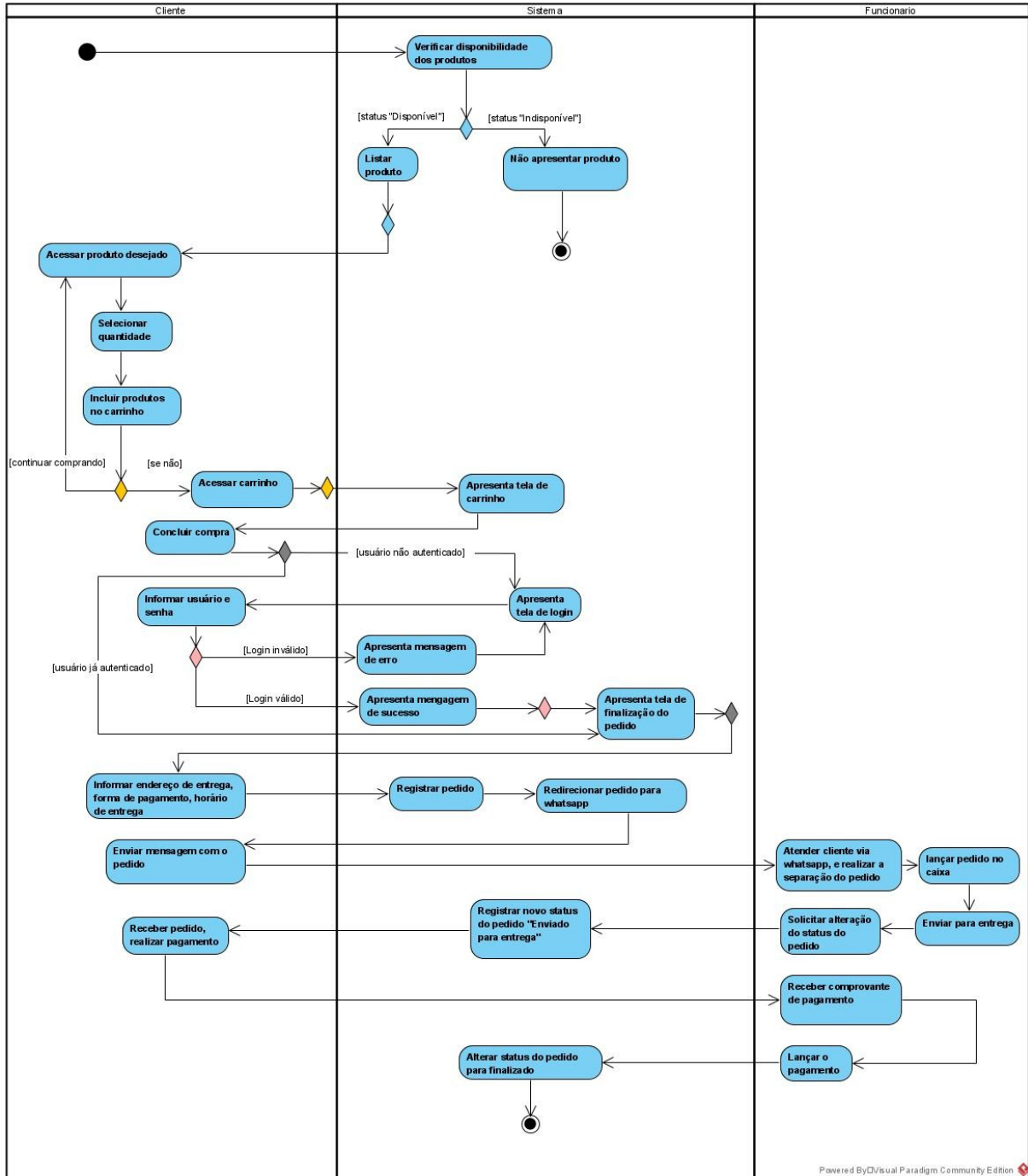


Figura 10 – Diagrama de atividades para o processo de Efetuar Pedido

A Figura 11 apresenta o diagrama de classes preliminar que representa as classes do banco de dados da aplicação.

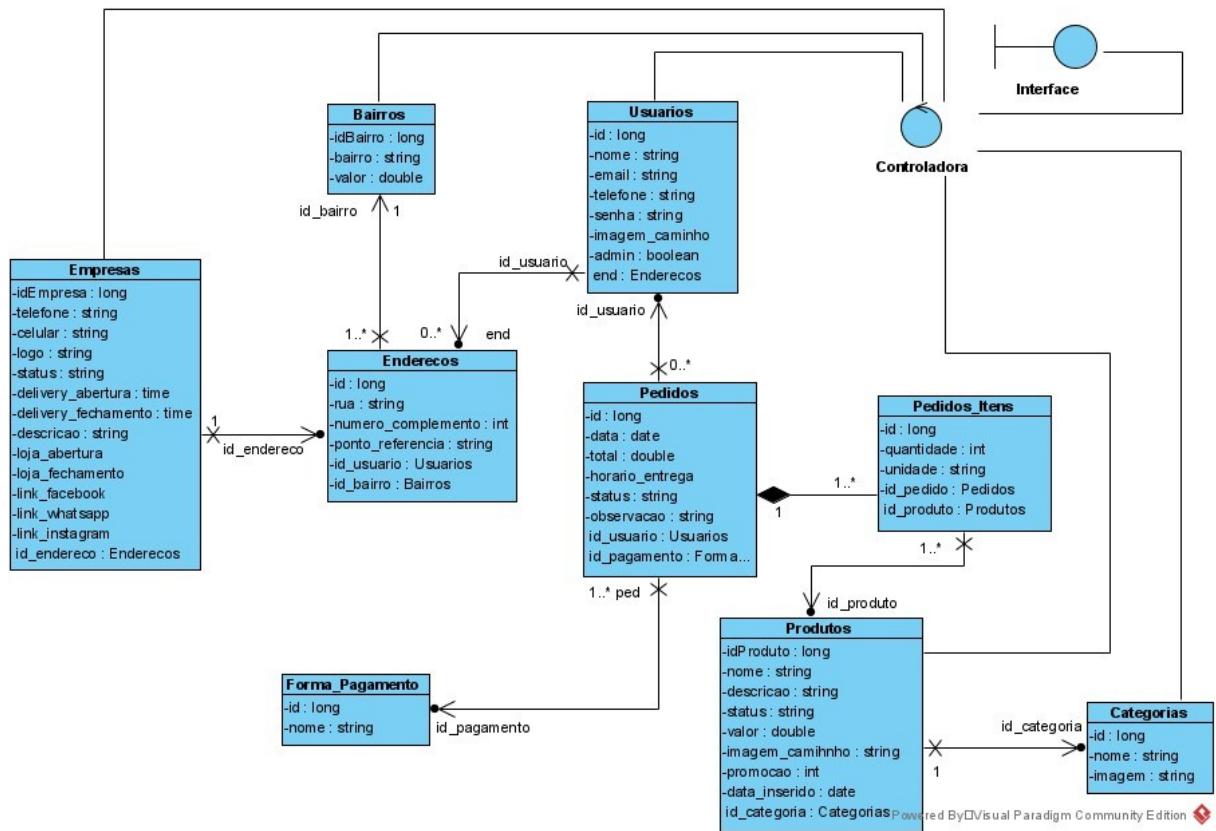


Figura 11 – Diagrama de classes preliminar

A Figura 12 apresenta o diagrama de entidades-relacionamentos (DER) que representa o banco de dados da aplicação.

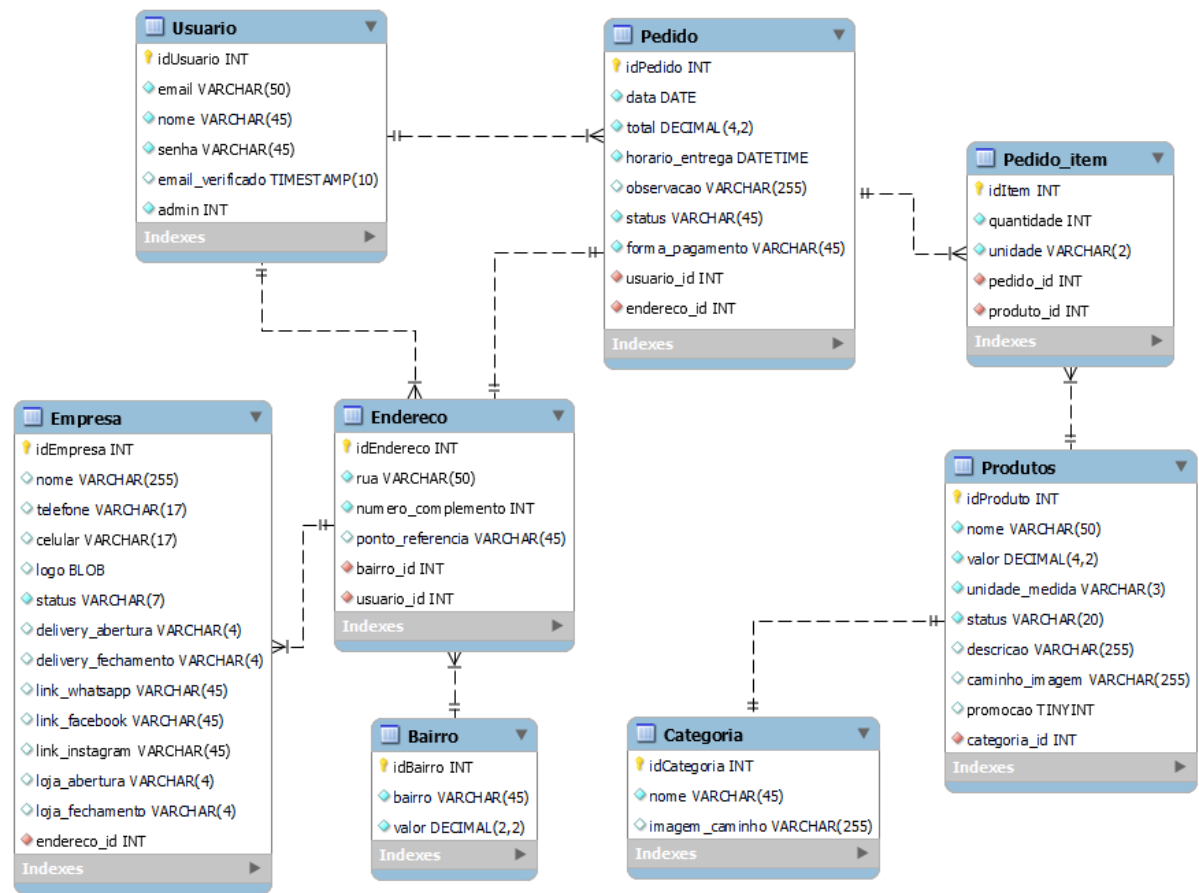


Figura 12 – Diagrama de entidade relacionamento

4.3 APRESENTAÇÃO DO SISTEMA

O desenvolvimento deste se deu como base no Laravel, um *framework* web baseado em PHP que é amplamente baseado na arquitetura MVC. Essa sigla se justifica, porque esse padrão de arquitetura separa um aplicativo em três componentes lógicos principais: o modelo, a visualização e o controlador. Essa separação de camadas ajuda na redução de acoplamento e promove o aumento de coesão nas classes do projeto. Assim, quando o modelo MVC é utilizado, pode facilitar a manutenção do código e sua reutilização em outros projetos. (MEDEIROS, 2013). Essa característica permite que o desenvolvimento do projeto fique mais organizado, separando as partes do projeto em módulos.

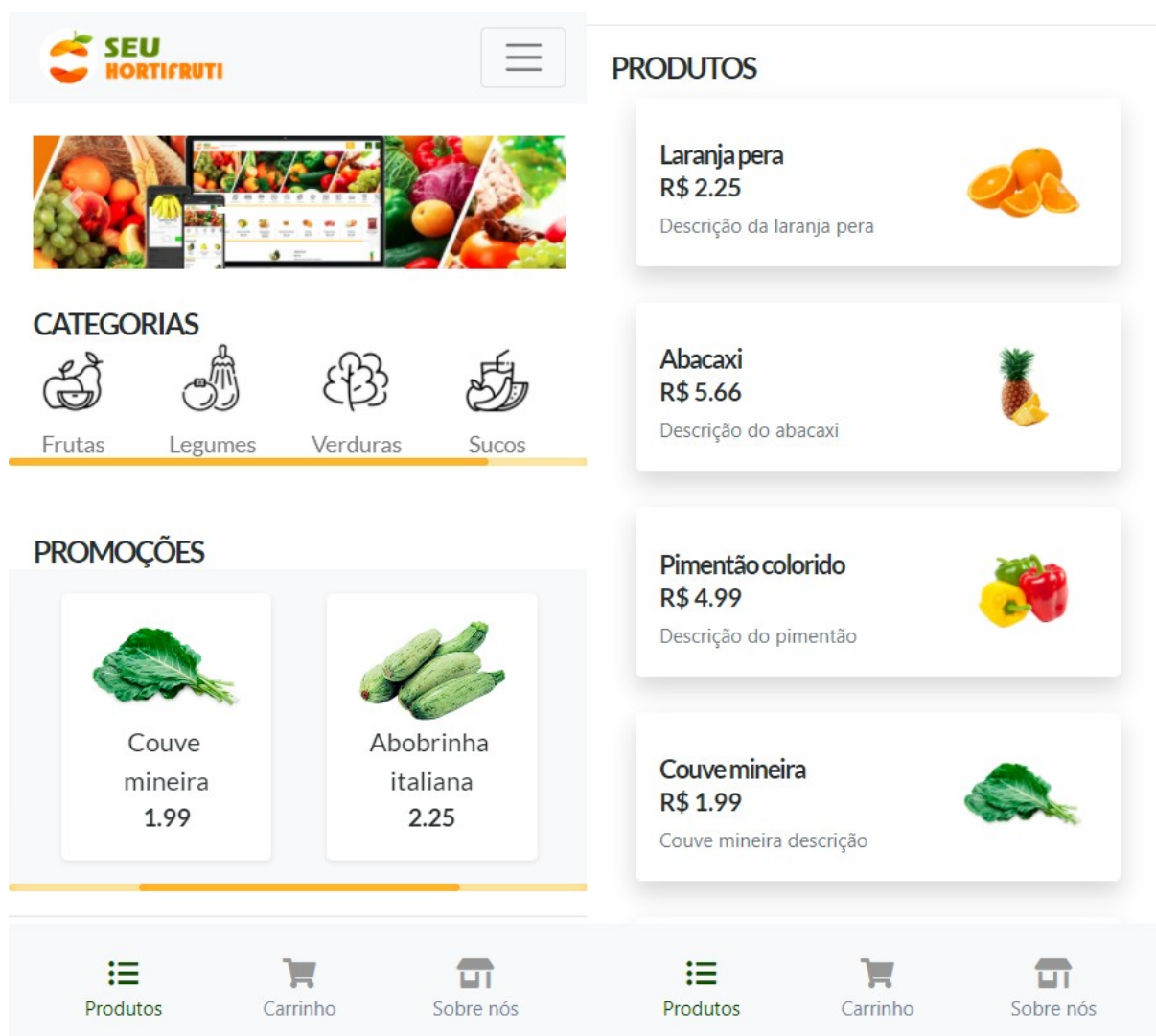


Figura 13 – Página inicial do aplicativo

A Figura 13 apresenta a aba produtos da página inicial do aplicativo que conta com um banner, busca por produtos e categorias e produtos em promoção. De acordo com a

arquitetura MVC, essa tela representa a camada de visão que tem como papel receber os dados do controle e exibir para o usuário. Apenas os produtos disponíveis para compra com *status* “ativo” aparecem nessa tela. Para o desenho do *layout* das telas do sistema, foram utilizados componentes do *Bootstrap 4* como: *dropdown*, *navbar*, *carousel*, *cards*, entre outros. Para implementação dos ícones foi utilizado o conjunto de ferramentas de fontes e ícones *Font Awesome*.

```
Route::get('/', function () {
    $categoria = Categoria::All();
    $produto = Produto::All();
    return view('welcome', ['produtos' => $produto, 'categorias' =>
    $categoria, 'produtosFiltrados' => $produto ]);
});
```

Listagem 1 – Rota para tela principal

No Laravel todas as solicitações são mapeadas com a ajuda de rotas. O roteamento básico envia a solicitação para os controladores associados ou direto para a camada de visualização. Quando o usuário acessar o link da aplicação, a rota descrita na Listagem 1 será invocada. O método *All()* recuperará todos os registros da tabela de banco de dados associada aos modelos, categorias e produtos redirecionando-os para a página inicial da aplicação (*welcome.blade.php*) na camada de visualização.

```

@foreach($produtosFiltrados as $produto)
    @if($produto->status==1)
        <div class="col-sm-12 col-lg-6 mt-2" data-toggle="modal"
            data-target="#modal-mensagem{{ $produto->id }}">
            <div class="card-editado mb-3 shadow p-3 bg-white rounded">
                <div class="row no-gutters">
                    <div class="col-8">
                        <div class="card-body-editado">
                            <h6 class="titulo-lista">{{ $produto-
>nome }}</h6>

                            <p class="card-text valor-lista">R$
{{ $produto->valor }}</p>

                            <p class="card-text"><small
                                class="text-muted">{{ $produto-
>descricao }}</small>

                                </p>
                            </div>
                        </div>
                    </div>
                    <div class="col-4 alinhar-imagem">
                        
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

Listagem 2 – Página inicial apresenta produtos

Blade é o mecanismo de modelagem que está incluído no Laravel. Todos os modelos Blade são compilados em código PHP simples, possibilitando o reuso de código e simplificando a inserção de trechos PHP em páginas HTML com uma sintaxe mais limpa. Logo, quando a página *welcome.blade.php* é chamada, é possível usar a estrutura de repetição através da anotação *@foreach* para apresentar todos os produtos e categorias na tela do usuário. A declaração condicional *@if* filtra os produtos que estão inativos, mostrando somente os ativos para o usuário. Para exibir o conteúdo da variável na tela, basta envolver a variável entre colchetes, como é o caso de `{{ $produto->nome }}`, que apresenta o nome do produto.

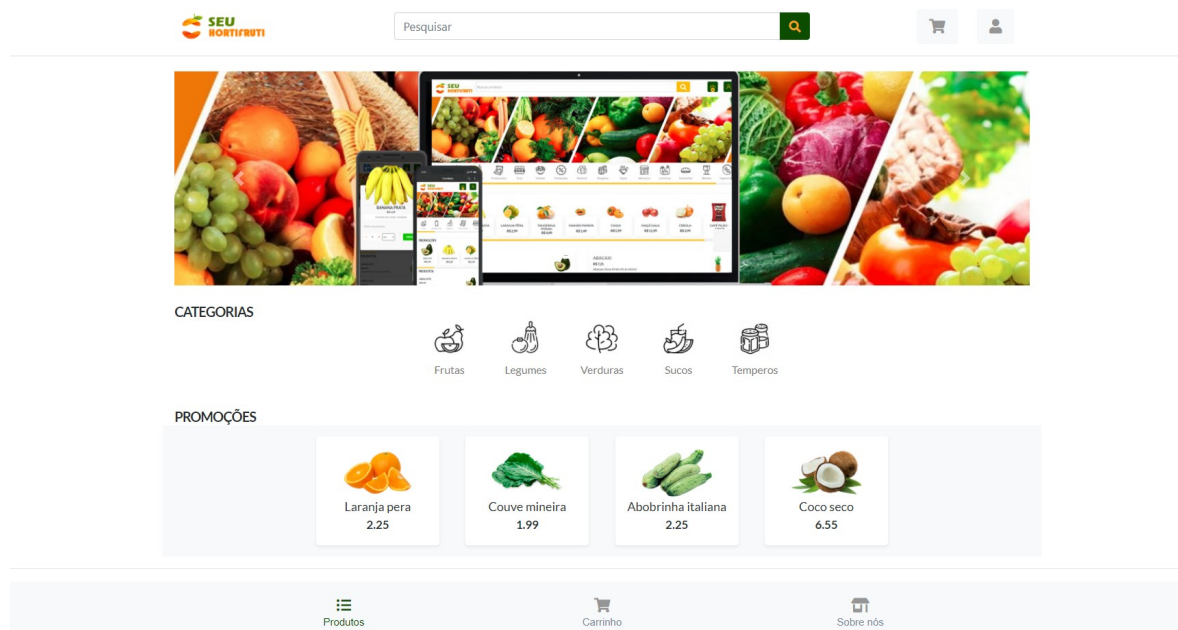


Figura 14 – Responsividade na página inicial

O sistema é uma aplicação web responsiva desenvolvida com a ferramenta *Bootstrap*, que inclui um sistema de grade móvel responsivo, o qual dimensiona conforme o tamanho do dispositivo ou janela de visualização aumenta. O sistema de grade do Bootstrap usa uma série de contêineres, linhas e colunas para fazer o layout e alinhar o conteúdo da página, adaptando-se ao tamanho da tela que o estiver acessando. A Figura 14 e 13 exemplificam esta responsividade apresentando a tela principal nas dimensões de um *desktop* com resolução de 1360x768 pixels.

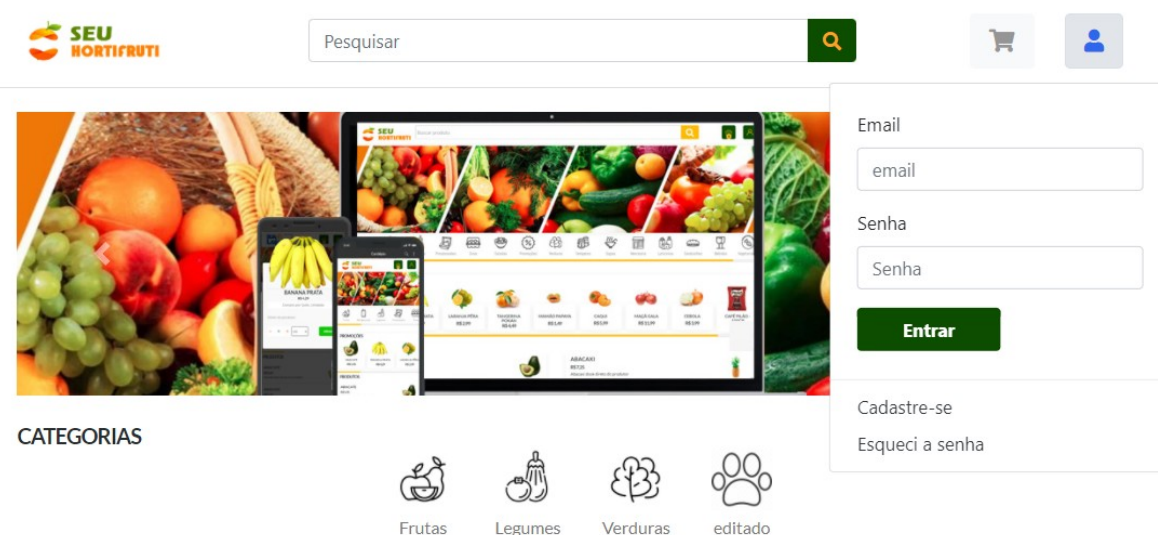


Figura 15 – Tela de login do usuário

A Figura 15 apresenta o modal de login do usuário, onde será validado o e-mail e senha do mesmo, sendo também possível realizar o cadastro para o uso da aplicação no primeiro acesso do usuário.

```

public function authenticate(Request $request){
    $credentials = $request->only('email', 'password');

    if (Auth::attempt($credentials)) {
        return redirect()->intended('/');
    } else{
        return redirect()->intended('/')->with('message', 'Usuário não
logou');
    }
}

```

Listagem 3 – Autenticação de usuários

A função apresentada na Listagem 3 é disparada ao clicar no botão “Entrar” na área de login, conforme apresentada na Figura 15. A função insere o e-mail e senha informado pelo usuário na variável *credentials* e em seguida, utiliza essa variável como parâmetro no método *attempt* fornecido pelo *framework* Laravel para lidar com a tentativa de autenticação. O Laravel inclui autenticação embutida e serviços de sessão que são tipicamente acessados através da classe *Auth*. Esses recursos fornecem autenticação baseada em *cookies* para solicitações iniciadas em navegadores da web.

The image shows a web application interface for user registration. At the top left is the logo for 'SEU HORTIFRUTI'. To its right is a search bar with the placeholder text 'Pesquisar' and a magnifying glass icon. Further right are icons for a shopping cart and a user profile. The central part of the page is a registration form titled 'Cadastro'. It contains several input fields: 'Nome*' and 'Celular*' (with a phone icon), 'Email*', 'Senha*' (password), and 'Repita a senha*' (confirm password). Below these is an 'Imagem' section with a 'Choose File' button and the text 'No file chosen'. A prominent green button labeled 'Registrar' is positioned at the bottom of the form. The footer of the page has three navigation links: 'Produtos' (with a list icon), 'Carrinho' (with a shopping cart icon), and 'Sobre nós' (with a house icon).

Figura 16 – Tela de cadastro de usuário

A Figura 16 apresenta a tela de cadastro de usuários, que contém informações pertinentes ao cliente. Os campos nome, celular, e-mail e senha são obrigatórios. Caso o usuário não insira imagem de perfil, o sistema vai definir uma imagem de perfil padrão para o usuário.

```

public function novoUsuario(Request $request){
    if($request->image){
        $request->validate([
            'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);
        $imageName = time().'.'.$request->image->extension();
        $request->image->move(public_path('images/usuarios'), $imageName);
    } else {
        $imageName = "default-user.png";
    }
    //verifica se todos campos estao inseridos
    if($request->nome && $request->telefone && $request->email && $request->password && $request->password2){
        //verifica se os campos de senha sao iguais
        if($request->password == $request->password2){
            //email e celular devem ser unicos
            $emailjaexiste = User::where('email', '=', $request->input('email'))->first();
            $celularjaexiste = User::where('email', '=', $request->input('telefone'))->first();
            if ($emailjaexiste === null && $celularjaexiste === null ) {
                User::create([
                    'nome' => $request->nome,
                    'email' => $request->email,
                    'telefone' => $request->telefone,
                    'password' => Hash::make($request->password),
                    'imagem_caminho' => $imageName,
                    'admin' => 0
                ]);
                return redirect()->back()->with('message', 'Usuario cadastrado com sucesso');
            } else {
                return redirect()->back()->with('warning', 'Email ou celular já está sendo utilizado por outra conta');
            }
        } else {
            return redirect()->back()->with('warning', 'Verifique sua senha, os dois campos de senha devem ser iguais');
        }
    } else {
        return redirect()->back()->with('warning', 'Por favor, preencha todos os campos solicitados');
    }
}

```

}

Listagem 4 – Cadastro de usuários

O trecho de código na Listagem 4 refere-se a função de criar um novo usuário na camada de controle. O processo se inicia na tela de cadastro referenciada na Figura 16, as informações do cliente são enviadas para essa função que realiza a verificação da imagem, verifica os campos obrigatórios, faz a consulta na base para verificar se já existe um e-mail ou celular cadastrado e, por fim, cria o usuário a partir do comando `User::create`. O Laravel usa um algoritmo de *hashing* denominado *Bcrypt* pelo uso do comando `Hash::make()`, o que significa que ele não salva a senha do usuário no banco de dados.

id	Imagem	Nome	Descrição	Valor	Unidade_medida	Categoria	Status	Promoção	Editar	Deletar
7		Abacaxi marataizes	Descrição do abacaxi	5.66	UN	Frutas	Ativo	Não		
8		Pimentão colorido	Pimentão colorido descrição	4.66	KG	Legumes	Ativo	Sim		
9		Couve mineira	Descrição da couve	1.99	UN	Frutas	Inativo	Sim		
10		Abobrinha italiana	Descrição da abobrinha	3.25	UN	Legumes	Ativo	Sim		
11		Maçã	descrição da maçã	2.99	KG	Frutas	Ativo	Não		
12		Pêra	descrição da pera	4.35	KG	Frutas	Ativo	Sim		

Figura 17 – Painel do administrador

A Figura 17 apresenta a tela de controle do administrador, onde apenas usuários do tipo administrador poderão acessar. Nessa tela é possível adicionar, ver, editar e remover produtos, categorias, bairros atendidos, usuários e pedidos. Também sendo possível editar as informações da empresa e gerar relatórios de venda.

```

public function paineladm(){
$usuario = User::find(Auth::id());
    if (Auth::check() && $usuario->admin==1){
        $bairros = Bairro::All();
        $categorias = Categoria::All();
        $produtos = Produto::All();
        $pedidos = Pedido::All();
    }
}

```

```

        $pedidoItens = PedidosItens::All();
        $usuarios = User::All();
        $empresa = Empresa::Find(1);
        return view('admin/admin', ['produtos' => $produtos, 'bairros' =>
        $bairros, 'pedidos' => $pedidos, 'pedidoItens' => $pedidoItens, 'usuarios' =>
        $usuarios, 'empresa' => $empresa], ['categorias' => $categorias]);
    } else{
        return redirect()->intended('/');
    }
}

```

Listagem 5 – Função que retorna o painel do administrador

A função descrita na Listagem 5 é chamada somente quando o usuário tenta acessar o painel do administrador pela rota “/painel-adm”. Para verificar se o usuário já está autenticado, o método de verificação *Auth::check()* é usado. Caso o usuário esteja conectado com perfil de administrador, então a tela do painel de administrador é carregada, caso contrário, a tela inicial *welcome.blade.php* é carregada. O *framework* Laravel tem uma autenticação de usuário segura e recursos de acesso restrito são fáceis de criar quando comparado com outros *frameworks* PHP.

```

<tbody>
    @foreach($produtos as $produto)
        <tr>
            <td>{{ $produto->id }}</td>
            <td></td>
            <td>{{ $produto->nome }}</td>
            <td>{{ $produto->descricao }}</td>
            <td>{{ $produto->valor }}</td>
            <td>{{ $produto->unidade_medida }}</td>
            <td>{{ $produto->id_categoria }}</td>
            <td>{{ $produto->status }}</td>
            <td>{{ $produto->promocao }}</td>
            <td>
                <a href="/editar-produto/{{ $produto->id }}"><i
                    class="fas fa-edit"></i></a></td>
            <td><a href="/deleta-produto/{{ $produto->id }}"><i
                    class="far fa-trash-alt"></i></a></td>
        </tr>
    @endforeach
</tbody>

```

Listagem 6 – Painel do administrador

A Listagem 6 refere-se a Figura 17, o código mostra como foi feita a apresentação de produtos na tela do painel do administrador. A partir de uma anotação `@foreach` na variável `produtos`, é possível consultar o id, imagem, nome, descrição, valor, unidade de medida, categoria, status e promoção de cada produto. Nas duas células finais da tabela há dois *links*, sendo o primeiro para editar e o segundo para excluir o produto.

The image shows a web application interface for product registration. The main window is titled "Cadastro de Produtos" and contains the following fields and controls:

- Nome:** A text input field.
- Categoria:** A dropdown menu with "Frutas" selected.
- Valor:** A text input field.
- Unidade de medida:** A dropdown menu with "KG" selected.
- Imagem:** A file upload field with a "Choose File" button and the text "No file chosen".
- Descrição:** A large text area for entering the product description.
- Ativo:** A checkbox.
- Promoção:** A checkbox.
- Registrar:** A green button to save the product.

The background shows a sidebar with navigation options: "Produtos", "Categorias", "Bairros", "Pedidos", "Usuarios", "Informações da empresa", and "Relatório de vendas". A table of products is visible on the right, with columns for "Editar" and "Deletar".

Figura 18 – Modal para cadastro de produto

A Figura 18 apresenta a janela para realizar o cadastro de um novo produto, sendo acionado ao clicar o botão com o ícone do sinal de “+” ao lado do título “Lista de Produtos” conforme exibido na Figura 17. Nele deve ser informado o nome, categoria, valor, unidade de medida, enquanto que os campos imagem, descrição e checkbox de ativo e promoção não são de preenchimento obrigatórios.

```

public function novoProduto(Request $request){
    $request->validate([
        'image' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
    ]);

    $imageName = time().'.'.$request->image->extension();

    $request->image->move(public_path('images/produtos'), $imageName);

    Produto::create([
        'nome' => $request->nome,
        'descricao' => $request->descricao,
        'status' => $request->status,
        'valor' => $request->valor,
        'unidade_medida' => $request->unidade_medida,
        'promocao' => $request->promocao,
        'id_categoria' => $request->id_categoria,
        'imagem_caminho' => $imageName
    ]);

    return redirect()->back()->with('message', 'Produto cadastrada com
sucesso');
}

```

Listagem 7 – Cadastro de produtos

A Listagem 7 refere-se à função *novoProduto()* na camada controladora de cadastro de produto. Sendo disparada ao clicar no botão “Registrar” mostrado na figura 18. Essa função realiza a validação da imagem e a armazena em uma pasta /produtos no servidor. A inserção do novo produto no banco de dados é feita através do *Eloquent*, um mapeador objeto-relacional (ORM) que realiza a interação com o banco de dados. Ao usar o *Eloquent*, cada tabela do banco de dados tem um "Modelo" correspondente que é usado para interagir com aquela tabela. Logo, através do método “Create” realizado no modelo “Produto”, é possível lançar as informações na tabela produtos.

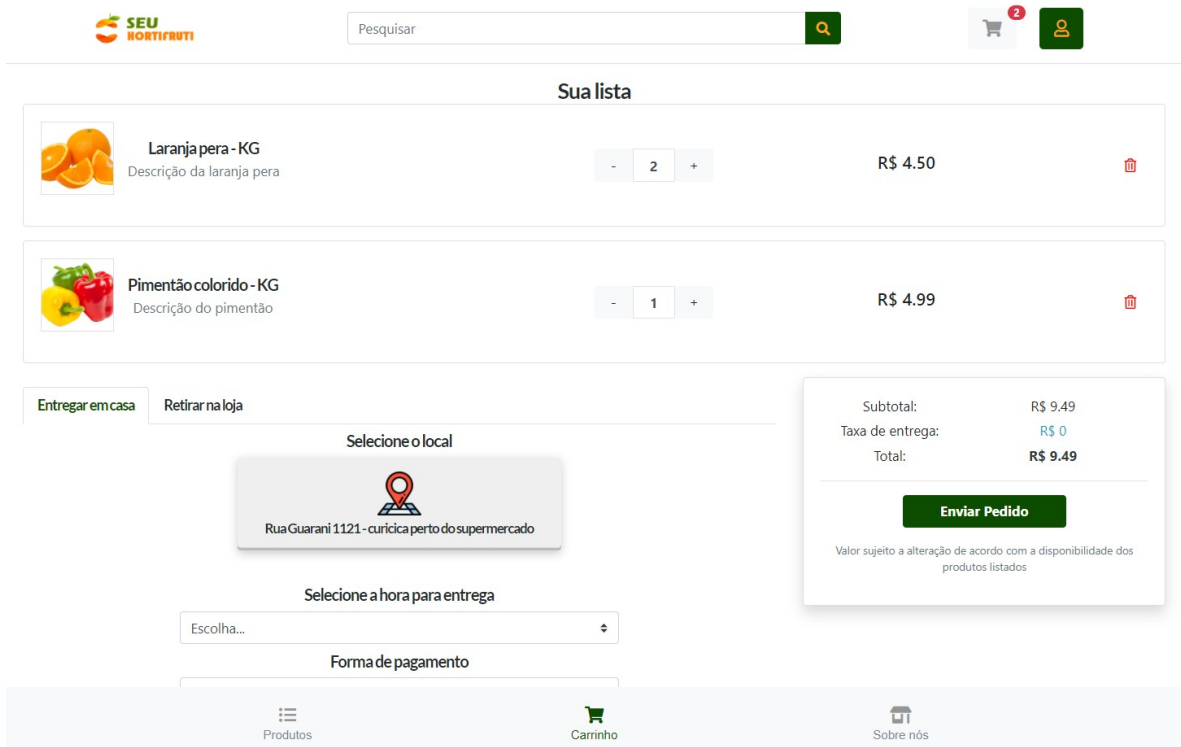


Figura 19 – Tela do carrinho de compras

A Figura 19 apresenta a tela do carrinho de compras com dois itens adicionados. Sendo possível editar a quantidade e remover os produtos incluídos. As informações foram organizadas para exibir a visualização do subtotal, a taxa de entrega, o valor unitário e o total. O cliente tem a opção de retirar o pedido na loja ou receber em um dos endereços cadastrados por ele. Para finalizar o pedido, o usuário deve selecionar o local, hora para entrega e forma de pagamento. O campo “Observação” não é de preenchimento obrigatório, porém caso seja selecionada a forma de pagamento “Dinheiro”, o usuário deverá informar o valor a ser pago maior ou igual ao valor total.

```
public function addToCart($id, Request $request){
    $product = Produto::find($id);
    if(!$product) {abort(404);}
    $cart = session()->get('carrinho');
    $quantidade = $request->input('quantidade');
    $unidade = $request->get('unidade');
    $valor = number_format($quantidade * $product->valor, 2, '.', '');
    // primeiro produto
    if(!$cart) {
        $cart = [
            $id => [
                "id" => $product->id,
                "nome" => $product->nome,
```

```

        "quantidade" => $quantidade,
        "valor" => $valor,
        "descricao" => $product->descricao,
        "imagem" => $product->imagem_caminho,
        "unidade" => $unidade
    ]
];
session()->put('carrinho', $cart);

$this->totalCart($id);//valores
return redirect()->back()->with('message', 'Produto adicionado ao
carrinho!');
}
// carrinho não esta vazio incrementa
if(isset($cart[$id])) {
    $cart[$id]['quantidade'] = $cart[$id]['quantidade'] + $request-
>input('quantidade');
    $cart[$id]['valor'] += $valor;
    session()->put('carrinho', $cart);

    $this->totalCart($id);//valores
    return redirect()->back()->with('message', 'Produto adicionado ao
carrinho!');
}
}
}

```

Listagem 8 – Controle do carrinho de compras

A Listagem 8 faz referência à função de adicionar um item no carrinho localizada na camada de controle do carrinho. O comando *Produto::find(\$id)* recupera o produto com o *id* informado na base de dados e o armazena na variável *\$product*. Foram usadas seções para armazenamento dos produtos no carrinho, permitindo uma maneira de armazenar informações sobre os produtos localmente. O comando *\$cart = session()->get('carrinho')* recupera o vetor de produtos “carrinho” na seção, para depois verificar se o mesmo existe. As variáveis *\$quantidade* e *\$unidade* são informadas pelo usuário ao passo que a variável *\$valor* é obtida a partir da multiplicação da quantidade pelo valor do produto, tendo seu resultado formatado para possuir duas casas decimais após a vírgula. Em seguida, caso o vetor de produtos carrinho seja *null*, para armazenar dados na sessão, o método *put()* da instância de solicitação é usado em *session()->put('carrinho', \$cart)*. Caso o carrinho já esteja com o produto adicionado, somente são editados os campos valor e quantidade na seção.

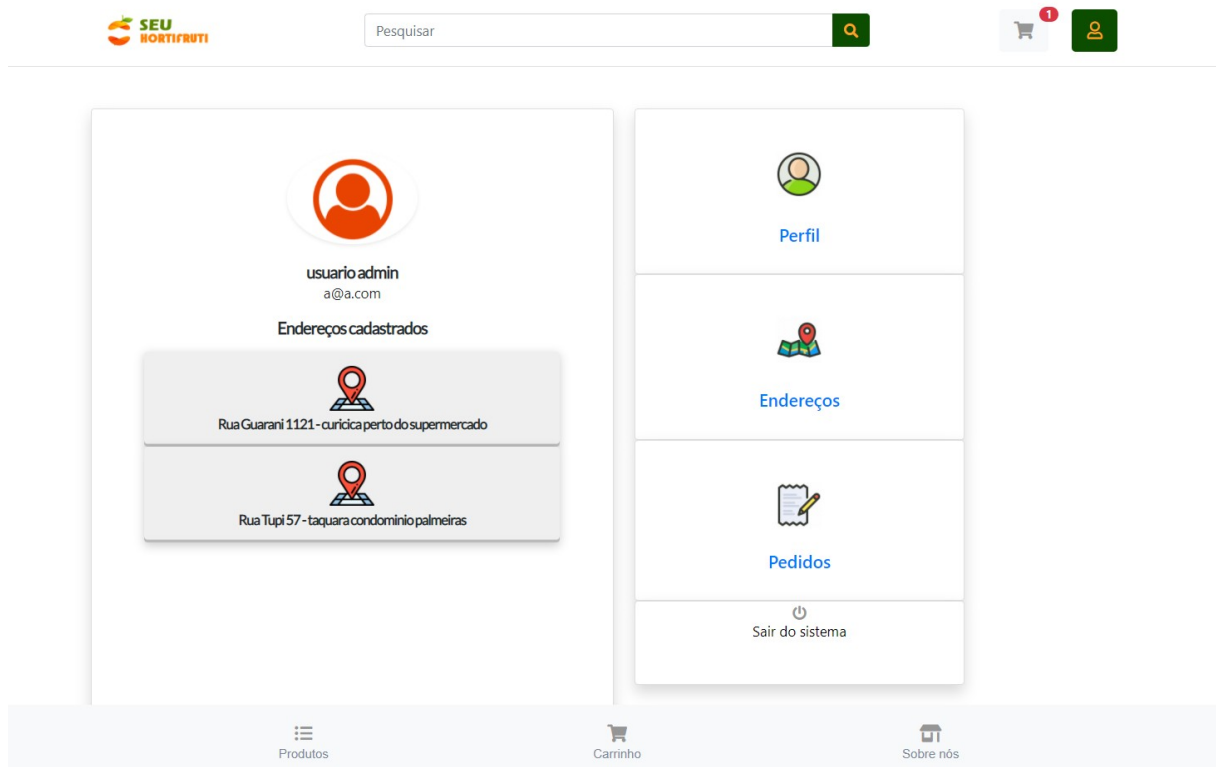


Figura 20 – Tela perfil do usuário

A Figura 20 apresenta a tela de perfil do usuário, nessa tela o usuário do sistema poderá editar seus dados, cadastrar endereços, verificar seu histórico de pedidos e sair da sua conta. Os endereços cadastrados aparecerão para seleção na tela do carrinho de compras (Figura 19).

```
public function efetuaPedido(Request $request){
    $cart = session()->get('carrinho');
    $total = session()->get('total');
    $obs = $request->observacao;
    $data = Carbon::now();
    $endereco = $request->endereco;

    $pedido = Pedido::create([
        'data' => $data,
        'total' => $total['total'],
        'horario_entrega' => $request->horario_entrega,
        'observacao' => $obs,
        'status' => "Pedido realizado",
        'id_endereco' => $endereco,
        'forma_pagamento' => $request->forma_pagamento,
        'id_usuario' => Auth::id()
    ]);
}
```

```

foreach ($cart as $produto) {
    PedidosItens::create([
        'unidade' => $produto['unidade'],
        'valor' => $produto['valor'],
        'quantidade' => $produto['quantidade'],
        'id_pedido' => $pedido->id,
        'id_produto' => $produto['id']
    ]);
}

session()->forget('carrinho', $cart);
session()->forget('total', $total);

return $this->enviaWhatsapp();
}

```

Listagem 9 – Controle do carrinho de compras

A Listagem 9 tem como função a finalização do pedido, é chamada quando o usuário clica em no botão “Enviar Pedido” na tela do carrinho de compras e todos os campos obrigatórios foram preenchidos conforme apresentado na Figura 19.

A função consulta as variáveis de seção “carrinho” e “total”, identifica o endereço informado e a observação descrita pelo cliente caso exista, e armazena a data atual utilizando a classe *Carbon* que é herdada da classe *PHP DateTime*. Assim que as variáveis são inseridas, é criado o registro do pedido na tabela “Pedidos” e cada produto selecionado pelo cliente na tabela “Pedidos_itens”. Após a inserção na tabela, O método *forget()* irá remover os dados da sessão referentes ao carrinho e total para zerar o carrinho do usuário. Por fim será chamada a função *enviaWhatsapp()* para enviar a mensagem do pedido pelo *Whatsapp*.

The image shows a WhatsApp chat interface. At the top, there's a green header with the WhatsApp logo and navigation links: WHATSAPP WEB, FEATURES, DOWNLOAD, SECURITY, HELP CENTER. Below the header, it says "Chat on WhatsApp with +55 46 99937-1096" and a green button labeled "CONTINUE TO CHAT".

The main chat area contains a message with the following text:

PEDIDO Nome: *teste* ----- *ITENS* 2 KG Laranja pera 1 UN Abacaxi 2 KG Pimentão colorido ----- Forma entrega: *DELIVERY* ----- *Condições de Pagamento* Forma de pagamento: *_Débito*_ ----- *Endereço de Entrega* Rua: *Rua Guarani* Número: *1121* Bairro: *curicica* Valor Entrega: *R\$ 5.00 ----- Sacolas: _____ Assinatura: _____ Aguarde enquanto separamos seu pedido. Acesse nossas redes sociais: *@seuhortifruti* Obrigado Pela Preferência!

Below the message, it says "Don't have WhatsApp yet?" with a "Download" link.

On the right side, there's a sidebar with the following details:

PEDIDO
Nome: **admin**

ITENS
2 KG Laranja pera
1 UN Abacaxi
2 KG Pimentão colorido

Forma entrega: **DELIVERY**

Condições de Pagamento
Forma de pagamento: **Débito**

Endereço de Entrega
Rua: **Rua Guarani**
Número: **1121**
Bairro: **curicica**
Valor Entrega: *R\$ 5.00

Sacolas: _____

Assinatura: _____

Aguarde enquanto separamos seu pedido.

Acesse nossas redes sociais:
@seuhortifruti

Obrigado Pela Preferência! 14:50 ✓

Figura 21 – Tela do *whatsapp* após a finalização do pedido

Após a finalização do pedido, o sistema redireciona o usuário para o *Whatsapp* da loja, através da API oferecida pelo *WhatsApp*, que permite gerar links diretos e iniciar uma conversa com a loja. A mensagem será formatada de acordo com o pedido do cliente contendo as seguintes informações: id do pedido, nome do cliente, itens do pedido (quantidade, unidade de medida e nome do produto), forma de entrega, forma de pagamento e endereço (rua, número, bairro e valor de entrega).

```

public function enviawhatsapp(){
    $empresa = Empresa::find(1);
    $pedido = Pedido::where('id_usuario', '=', Auth::id()->get()->last());
    $pedidoItens = PedidosItens::where('id_pedido', '=', $pedido->id)->get();

    $mensagem = 'https://api.whatsapp.com/send/?phone='. $empresa-
>celular . '&text=%2APEDIDO'. $pedido->id. '%2A%0ANome%3A+%2Ateste%2A
%0A-----%0A%2AITENS%2A%0A';

    foreach($pedidoItens as $pedidoItem){
        $mensagem .= $pedidoItem['quantidade'] . ' ' .
$pedidoItem['unidade'] . ' ' . Produto::find($pedidoItem['id_produto'])->nome .
'%0A';
    }

    $mensagem .= '-----%0AForma+entrega%3A+%2ADELIVERY%2A
%0A-----%0A%2ACondi%3%A7%3%B5es+de+Pagamento%2A
%0AForma+de+pagamento%3A+_%2A'. $pedido->forma_pagamento . '%2A_
%0A-----%0A%2AEndere%3%A7o+de+Entrega%2A%0ARua%3A+%2A' .
Endereco::find($pedido['id_endereco'])->rua . '%2A%0AN%3BAmero%3A+%2A' .
Endereco::find($pedido['id_endereco'])->numero_complemento . '%2A%0ABairro%3A+%2A'.
Bairro::find(Endereco::find($pedido['id_endereco'])->id_bairro)->nome . '%2A
%0AValor+Entrega%3A+%2AR%24+'.
Bairro::find(Endereco::find($pedido['id_endereco'])->id_bairro)->valor.
'%0A-----%0A%0ASacolas%3A+_____ %0A
%0AAssinatura%3A+_____ %0A
%0AAguarde+enquanto+separamos+seu+pedido.%0A%0AAcesse+nossas+redes+sociais%3A%0A
%2A%40seuhortifruti%2A%0A%0A0brigado+Pela+Prefer%3AAncia%21%0A&app_absent=0';

    return redirect()->away($mensagem);
}

```

Listagem 10 – Controle do carrinho de compras

A Listagem 10 tem a função de enviar o pedido para o *whatsapp* da loja. O envio da mensagem se dá com a criação de uma variável mensagem, a qual recebe a URL do *whatsapp*, acrescida do celular da empresa, cadastrado no painel administrador referenciado na Figura 14. Em seguida essa mensagem recebe todos os itens do pedido, para então ser adicionada a forma de pagamento e endereço. Por fim, o sistema redireciona o usuário para o domínio descrito no conteúdo da mensagem por meio do comando *redirect()->away(\$mensagem)*. Como resultado final o usuário enviará a mensagem no formato adequado, conforme mostra a Figura 21.

SEU HORTIFRUTI

Pesquisar

Informações da empresa

Nome da empresa
Seu Hortifruti

Endereço
Estr. da Água Grande

Delivery abre
09:00

delivery fecha
19:30

Telefone
46999371094

Celular
46999371094

Loja abre
07:30

Loja fecha
20:00

Logo
Choose File No file chosen

Imagem do banner 1
Choose File No file chosen

SEU HORTIFRUTI

Produtos

Carrinho

Sobre nós

Figura 22 – Tela dados da empresa

A Figura 22 apresenta a tela de cadastro e edição dos dados da empresa, localizada no painel do administrador. Essa tela permite a comunicação com a tabela “Empresa”, permitindo assim a alteração das seguintes informações da loja: Nome, endereço, horário de funcionamento, telefone, celular, logo, banners e links das redes sociais.

SEU HORTIFRUTI

Seu hortifruti
Av. Vicente de Carvalho, 909 - Vila da Penha

LOJA - HORÁRIO
07 as 20 - de segunda à sábado
07:30h as 14h - domingo

DELIVERY - HORÁRIO
09h as 19:30h - de segunda à sábado
09h as 13h - domingo

FACEBOOK

INSTAGRAM

WHATSAPP

Visualizar mapa ampliado

Academia New
Vila da Penha
Acquarone
Crossfit Netuno
Creche Ninho de
Amor Anália Fran
P. Tejuap

SEU HORTIFRUTI

VISA, Mastercard, elo, AMEX, Hipercard, JCB

Glêco, Glêco, VR, VR

Produtos, Carrinho, Sobre nós

Figura 23 – Tela sobre nós

A Figura 23 apresenta a tela “Sobre nós”, na qual são apresentados os dados da empresa cadastrada no painel do administrador, conforme mostrado na Figura 22.

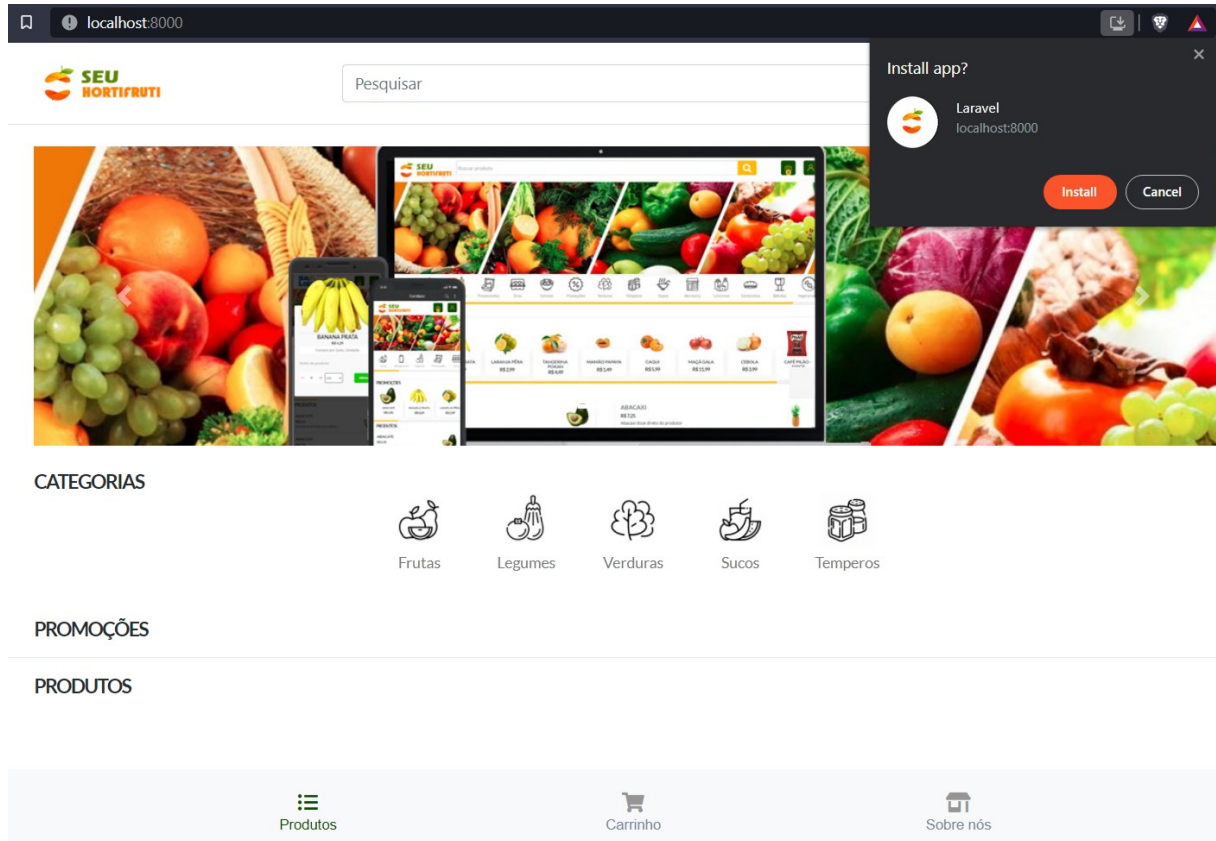


Figura 24 – Instalação PWA

A Figura 24 apresenta a capacidade de adicionar a aplicação à tela inicial, sendo uma das principais características de uma PWA, pois permite uma experiência de aplicativo nativo, que inclui a inicialização de aplicativos a partir da tela inicial do usuário. São considerados os recursos necessários para fazer um PWA: HTTP, um arquivo de manifesto e *service workers*. Para o propósito de testes, o ambiente *localhost* é apresentado na Figura 24 como um contexto seguro sem a necessidade de HTTPS.

```

var staticCacheName = "pwa-v" + new Date().getTime();
var filesToCache = [ /offline', '/css/app.css', '/js/app.js',
'/images/icons/icon-192x192.png', '/images/icons/icon-512x512.png'];
self.addEventListener("install", event => {
  this.skipWaiting();
  event.waitUntil(
    caches.open(staticCacheName)
      .then(cache => {
        return cache.addAll(filesToCache);
      })
  )
});
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames
          .filter(cacheName => (cacheName.startsWith("pwa-")))
          .filter(cacheName => (cacheName !== staticCacheName))
          .map(cacheName => caches.delete(cacheName))
      );
    })
  );
});
self.addEventListener("fetch", event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        return response || fetch(event.request);
      })
      .catch(() => {
        return caches.match('offline');
      })
  )
});
self.addEventListener('fetch', function(event) {});

```

Listagem 11 – Arquivo ServiceWorker.js

A Listagem 11 apresenta um dos arquivos necessários para se ter um PWA, o *Service Worker*. Este arquivo intercepta as requisições da aplicação, guardando os resultados no lado do cliente. Isso permite que a aplicação funcione no modo *offline*.

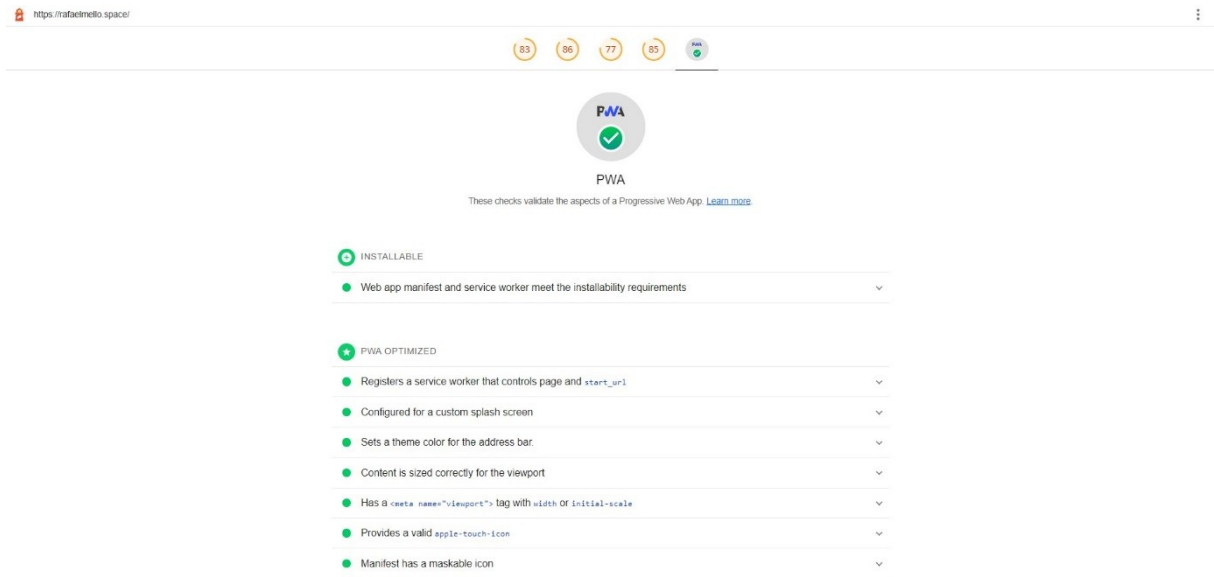
```

var staticCacheName = "pwa-v" + new Date().getTime();
var filesToCache = [ /offline', '/css/app.css', '/js/app.js',
'/images/icons/icon-192x192.png', '/images/icons/icon-512x512.png'];
self.addEventListener("install", event => {
  this.skipWaiting();
  event.waitUntil(
    caches.open(staticCacheName)
      .then(cache => {
        return cache.addAll(filesToCache);
      })
  )
});
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames
          .filter(cacheName => (cacheName.startsWith("pwa-")))
          .filter(cacheName => (cacheName !== staticCacheName))
          .map(cacheName => caches.delete(cacheName))
      );
    })
  );
});
self.addEventListener("fetch", event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        return response || fetch(event.request);
      })
      .catch(() => {
        return caches.match('offline');
      })
  )
});
self.addEventListener('fetch', function(event) {});

```

Listagem 12 – Arquivo Manifest.json

O manifesto do aplicativo da Web descrito na Listagem 12 é um arquivo JSON que fornece os metadados necessários para a PWA, permitindo se comportar de forma mais semelhante a um aplicativo nativo, instalável na tela inicial.



Listagem 25 – Relatório do Google Lighthouse

A figura 25 apresenta o relatório do Google Lighthouse, ferramenta de código aberto com função de medir a qualidade das páginas da web. O Google Lighthouse audita o desempenho, acessibilidade e otimização de mecanismo de pesquisa de páginas da web, informando ao usuário se o site se encaixa no padrão PWA.

4.4 CONCLUSÃO

Esse documento apresentou a proposta e implementação de um aplicativo web responsivo para a venda e *delivery* de produtos hortigranjeiros. Após realizar todas as etapas que fazem parte do processo de desenvolvimento de software, desde o levantamento de requisitos até a implementação e testes, o resultado foi uma aplicação *e-commerce* PWA com a possibilidade de atender as demandas para o serviço de entregas no setor de hortigranjeiros, mercearias e mercados.

A IDE de desenvolvimento foi o Visual Studio Code, essa ferramenta auxiliou o processo de desenvolvimento por ser leve, gratuita e fornecer suporte para tecnologias da Web como; HTML, CSS, JSON. Ainda conta com atalhos de teclado intuitivos, personalização fácil, e terminal embutido, o que auxilia nos comandos da interface (Artisan) por linha de comando usada no Laravel.

O *framework* Laravel, utilizado durante a codificação da aplicação, ofereceu uma ampla gama de vantagens em comparação com outros *frameworks* na linguagem de programação PHP. Isso porque combina agilidade de desenvolvimento web, codificação simples e um alto grau de customização. Por padrão, utiliza a arquitetura MVC no desenvolvimento, tendo por consequência um código mais organizado. Além de fornecer um grande número de bibliotecas pré-programadas, o que colabora com a agilidade de desenvolvimento neste *framework*.

A utilização de uma PWA, utiliza APIs e recursos do navegador para fazer um site multiplataforma ter o aspecto de um aplicativo e trazer a mesma experiência de um aplicativo nativo ao usuário. Sendo possível instalá-lo na tela inicial do celular, acessá-lo *offline* e receber notificações. Todas essas características são um benefício para os clientes, os quais não precisam instalar um aplicativo para fazer suas compras.

O *framework Bootstrap* foi de suma importância tanto para a responsividade do sistema quanto para a organização do layout das telas e agilidade de implementação. Isso pelo fato de oferecer diversas vantagens: ajustar o código para que tenha a mesma aparência em todos os navegadores, fornecer estruturas e estilos responsivas pré-programadas para o uso, possuir uma boa documentação e suporte da comunidade.

Durante o desenvolvimento do sistema, os conceitos como: Modelagem de diagramas, Análise de requisitos, Utilização do padrão de arquitetura de software MVC, Programação orientada a objetos, Programação Web foram solidificados e incrementados pelo acadêmico. A elaboração deste trabalho ajudou a colocar em prática conceitos-chave aprendidos durante o

Curso Análise e Desenvolvimento de Sistemas e que auxiliarão o acadêmico no futuro mercado de trabalho.

REFERÊNCIAS

ALDAY, H. E. C. PINOCHET, L. H. C. **A tecnologia e-commerce como estratégia determinante no setor supermercadista**. 5. ed. Curitiba: Rev. FAE, 2002. Disponível em: <https://revistafae.fae.edu/revistafae/article/view/482>. Acesso em: 15 abr. 2021.

ATER, T. **Building progressive web apps : bringing the power of native to the browser**. O'Reilly Media, 2017.

BOOCH G.; RUMBAUGH J.; JACOBSON I. **UML: guia do usuário**. Elsevier Brasil, 2006.

DA SILVA, B. P. **Understanding Customer Willingness to Wait (WTW) in the E-commerce Grocery Retail**. Faculdade de Engenharia da Universidade do Porto, 2018.

DONICI, A. N.; IGNAT I.; MAHA L. G. **E-commerce across United States of America: Amazon.com**. Economy Transdisciplinarity Cognition, v. 15, n. 1, 2012.

E-COMMERCE BRASIL. **E-commerce brasileiro cresce 73,88% em 2020, revela índice MCC-ENET**. Disponível em: <https://www.ecommercebrasil.com.br/noticias/e-commerce-brasileiro-cresce-dezembro> Acesso em: 5 abr. 2021.

FERREIRA DE ANDRADE, M. C.; GONÇALVES DA SILVA, N. T. **O COMÉRCIO ELETRÔNICO (E-COMMERCE): UM ESTUDO COM CONSUMIDORES**. Perspectivas em Gestão & Conhecimento, v. 7, n. 1, p. 98-111, 2017. Disponível em: <http://dx.doi.org/10.21714/2236-417X2017v7n1p98>. Acesso em: 15 abr. 2021.

FORTUNATO D.; BERNARDINO J. **Progressive web apps: An alternative to the native mobile Apps**. 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), 2018.

FRUTELLO. [Frutellodelivery.glideapp.io](https://frutellodelivery.glideapp.io). Disponível em: <https://frutellodelivery.glideapp.io>. Acesso em: 5 mai. 2021.

GOBACKLOG. **O Papel da Tecnologia na Fidelização de Clientes**. GoBacklog. Disponível em: <https://gobacklog.com/o-papel-da-tecnologia-na-fidelizacao-de-clientes/>. Acesso em: 25 maio 2021.

GOEL, R. **E-commerce**. New Age International Pvt, 2008.

HEILMANN C.;GOVE J.; CLARK S.; DUTTON S.; KURTULDU M.; WARRENDER R. **Progressive Web Apps: the future of the mobile web**. 5. ed. [s.l.]: Google, 2018 Disponível em: https://www.thinkwithgoogle.com/_qs/documents/6758/pwas-the-future-of-the-mobile-web.pdf. Acesso em: 14 abr. 2021.

HORTIFRUTI. [Delivery.hortifruti.com.br](https://delivery.hortifruti.com.br). Disponível em: <https://delivery.hortifruti.com.br/hortifruti>. Acesso em: 2 mai. 2021.

IFOOD. Disponível em: <https://shop.ifood.com.br>. Acesso em: 5 mai. 2021.

KLIZO. **Progressive Web Apps you use every day.** Medium. 2020. Disponível em: <https://medium.com/progressivewebapps/progressive-web-apps-you-may-use-every-day-164bfa92c498>. Acesso em: 1 mai. 2021.

LEE J., KIM H., PARK J., SHIN I. SON S. **Pride and Prejudice in Progressive Web Apps.** Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, p. 1731, 2018.

LOVE C. **Twitter Announces Their Progressive Web App (PWA) is in the Windows Store.** Love2dev.com. Disponível em: <https://love2dev.com/blog/twitter-pwa-windows-store/>. Acesso em: 2 mai. 2021.

MANZOOR A. **E-commerce: An introduction.** [s.l.]: Lambert Academic Publishing, 2010.
MENDES L. Z. R. E-commerce: origem, desenvolvimento e perspectivas. Trabalho de Graduação, Universidade Federal Do Rio Grande Do Sul, 2013.

MEDEIROS H. **Introdução ao Padrão MVC: Primeiros passos na Arquitetura MVC.** [online] Disponível em: <https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>. Acesso em: 24 nov. 2021.

MENDONÇA H. G. D. **E-commerce.** Revista Inovação, Projetos e Tecnologias, v. 4, n. 2, p. 240-251, 2016. Disponível em: <https://periodicos.uninove.br/ipotec/article/view/9361>. Acesso em: 29 abr. 2021.

MORTIMER G.; HASAN, S. F; ANDREWS, L.; Martin, J. **Online grocery shopping: the impact of shopping frequency on perceived risk.** The International Review of Retail, Distribution and Consumer Research, v. 26, n. 2, p. 202-223, 2016.

NAKAMURA, R. R. **E-commerce na Internet: Fácil de Entender.** São Paulo, Érica, ISBN 1970- 85-7194-750-3. 2001.

ORGÂNICOS IN BOX. [Organicosinbox.com.br](https://organicosinbox.com.br). Disponível em: <https://organicosinbox.com.br>. Acesso em: 5 mai. 2021.

OSMANI A. **Getting Started with Progressive Web Apps.** Google Developers. 2015 Disponível em: <https://developers.google.com/web/updates/2015/12/getting-started-pwa>. Acesso em: 14 abr. 2021.

PAN S.; GIANNIKAS V.; HAN Y.; GROVER-SILVA E.; QIAO B. **Using customer-related data to enhance e-grocery home delivery.** Industrial Management & Data Systems, v. 117, n. 9, p. 1917-1933, 2017. Disponível em: <https://www.emerald.com/insight/content/doi/10.1108/IMDS-10-2016-0432> Acesso em: 30 abr. 2021.

PUNAKIVI M.; SARANEN J. **Identifying the success factors in e-grocery home delivery.** International Journal of Retail & Distribution Management, v. 29, n. 4, p. 156-163, 2001.
RAMOS, E.; ANTUNES A., DO VALLE A. et al. E-commerce. 3. ed. Rio de Janeiro: Editora FGV, 2011.

REICHHELD F. F.; TEAL, T. **The Loyalty Effect The Hidden Force Behind Growth, Profits and Lasting Value**. Boston: Harvard Business School Press, 1996.

SCHIEFFELBEIN I.; MARTINS A. C. C.; FURIAN N. G. **Neoconsumidor e o Comportamento com Relação Ao Varejo Virtual**. [s.l.]: SEGeT, 2011. (VIII SEGeT – Simpósio de Excelência em Gestão e Tecnologia). Disponível em: <https://www.aedb.br/seget/arquivos/artigos11/57514722.pdf>. Acesso em: 5 abr. 2021.

SILVA NETO A. **Desenvolvimento de uma aplicação PWA que comporte outras aplicações usando arquitetura de micro frontend**. Pontifícia Universidade Católica de Goiás, 2020.

SINGH, REEMA. **Why do online grocery shoppers switch or stay? An exploratory analysis of consumers' response to online grocery shopping experience**. International Journal of Retail & Distribution Management, v. 47, n. 12, p. 1300-1317, 2019. Disponível em: <https://www.emerald.com/insight/content/doi/10.1108/IJRDM-10-2018-0224>. Acesso em: 3 abr. 2021.

SOUZA R. **6 Exemplos de PWAs bem sucedidos | Prox**. Prox. 2021. Disponível em: <https://prox.com.br/6-exemplos-de-pwas-bem-sucedidos>. Acesso em: 1 mai. 2021.

TRINDADE P. E.; AFFINI L. P. **Apontamento a cerca do progressive web apps**. Jornada Internacional GEMInIS, v. III, p. 4, 2018. Disponível em: <https://doity.com.br/media/doity/submissoes/artigo-6d73074b34b7631db712c0472463b10b08ed8ea4-arquivo.pdf>. Acesso em: 14 abr. 2021.