

Desenvolvimento do sistema de catálogo de produtos: empresa Sneakers

Bruno Pinheiro Vidal, Gutelvam Rodrigues de Jesus, Jefferson John Parnaíba Dias, Kaio Saíum Teoi, Leonardo Távora Auad¹

Orientadores: Professora Me. Simone de Abreu e Professor Esp. Victor Williams Stafusa da Silva

Faculdade Impacta de Tecnologia
São Paulo, SP, Brasil
10 de junho de 2021

Resumo. Foram evidenciados os métodos recorrentes de desenvolvimento, tais como o diagrama do fluxo de dados, os requisitos, as restrições e as regras de negócio. Todas as funcionalidades, tais como suas telas, são apresentadas de acordo com a visão dos *stakeholders*, e todas as ferramentas, *frameworks*, banco de dados e a linguagem usada para desenvolver o projeto são sucintamente explicadas, e todas possuem papéis muito importantes a serem desempenhados no desenvolvimento da plataforma.

Palavras-chaves: Relatório final, Engenharia de *Software*, Arquitetura de Sistema, Sistema de catálogo de produtos.

1. Introdução

Neste projeto, foi desenvolvida uma aplicação *web* com base nas necessidades do cliente, Sneakers, de oferecer a seus usuários uma interface de compras *online*, intuitiva e fácil de ser usada. Além disso, também foi disponibilizada a capacidade do cliente de controlar e monitorar suas vendas e mudanças constantes em seu estoque, de acordo com as compras feitas pelos usuários.

1.1. Apresentação da empresa Sneakers

A Sneakers é uma loja de artigos esportivos especializada em tênis colecionáveis, com reconhecimento notório por trazer tanto novidades como raridades, sempre com a melhor qualidade em produtos e preço justo, que reflete a excelência do serviço prestado.

Começou com um pequeno box de tênis importados, de marcas Premium mundiais, rapidamente cresceu como empresa na região, por trazer exclusivamente os produtos de maior qualidade, com pedidos dos clientes em tênis colecionáveis viu uma nova clientela surgindo, com margens de lucro ainda maior que alavancou o faturamento, vendendo frequentemente exemplares no patamar de 50 mil reais (Nike Air Jordan V Retro), em alguns produtos raríssimos o preço chega à casa de 150 mil reais (Nike Air Jordan III de 1988), alcançou um faturamento mensal médio hoje de 1 milhão de reais.

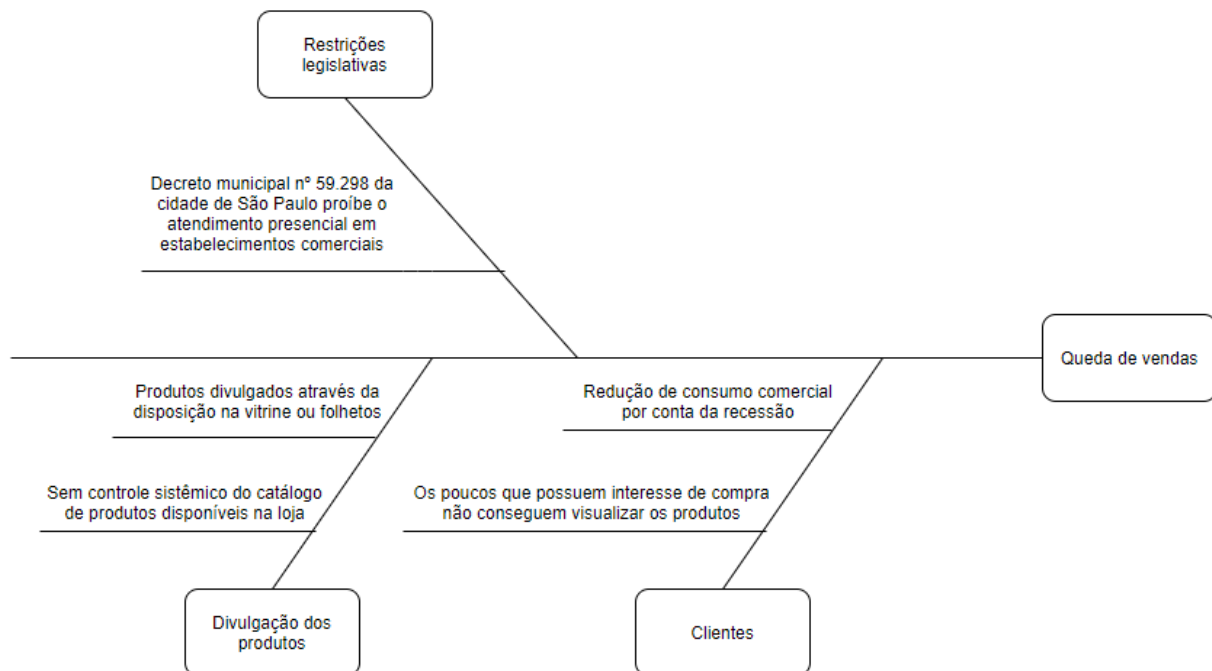
¹ Os autores podem ser contatados respectivamente pelos seus correios eletrônicos:

bruno.vidal@aluno.faculdadeimpacta.com.br, gutelvam.jesus@aluno.faculdadeimpacta.com.br, jefferson.dias@aluno.faculdadeimpacta.com.br, kaio.teoi@aluno.faculdadeimpacta.com.br, leonardo.auad@aluno.faculdadeimpacta.com.br.

1.2. O problema da crise pandêmica na empresa Sneakers

Com 20 anos de atuação no coração da capital metropolitana de São Paulo, situada na Avenida Paulista nº 175, possui 50 colaboradores, hoje enfrenta uma crise econômica devido a pandemia de COVID-19, que diminuiu a projeção de vendas físicas em 2020, por não ter um site próprio de vendas, necessita de atualização para plataforma digital, desejando a elaboração de um catálogo de produtos, tanto para visibilidade e manutenção de produtos por parte de gerência, quanto para os funcionários na realização de consultas e validação das descrições dos itens. Ao se adaptar aos meios virtuais de venda, a Sneakers busca retomar o ritmo de vendas durante a crise econômica.

Figura 1 – Diagrama de Ishikawa sobre o problema de vendas da Sneakers.



Fonte: Os autores.

1.3. Stakeholders

- Administradores: responsáveis pelo gerenciamento e atualização de produtos, tais como suas características, sendo elas disponibilidade, compras realizadas, descrição do produto, inserir e/ou remover produtos.
- Vendedores: utilizam da plataforma para vender seus produtos, disponibilizam os produtos a serem vendidos e atualizam seus preços conforme a necessidade do mercado.
- Estoquistas: se encontram na fronteira externa da aplicação. São responsáveis por realizar mudanças no estoque de acordo com as atualizações e compras realizadas.
- Usuários: aqueles que irão utilizar a plataforma para comprar os produtos.

1.4. Restrições

Tabela 1 – Tabela de restrições.

| Restrição | Razão lógica |
|---|--|
| A linguagem JavaScript deve ser utilizada | É a linguagem mais utilizada para os padrões <i>web</i> . |
| Orçamento de pessoal e equipamentos | Reflexo da pandemia a receita da empresa diminuiu e assim tem que ser o mais barato possível sem perder a qualidade. |
| Navegadores | Deve ter compatibilidade e retrocompatibilidade com os navegadores mais populares do mercado. |
| Banco de dados | Deve ser <i>open source</i> e deve ser utilizado em sistema windows. |
| Licenciamento da solução | Deve ser de renovação anual. |
| LGPD | Deve estar de acordo com as normas da LGPD. |
| Política Departamental | Devem ser criadas visões e acessos diferentes a depender do cargo. |

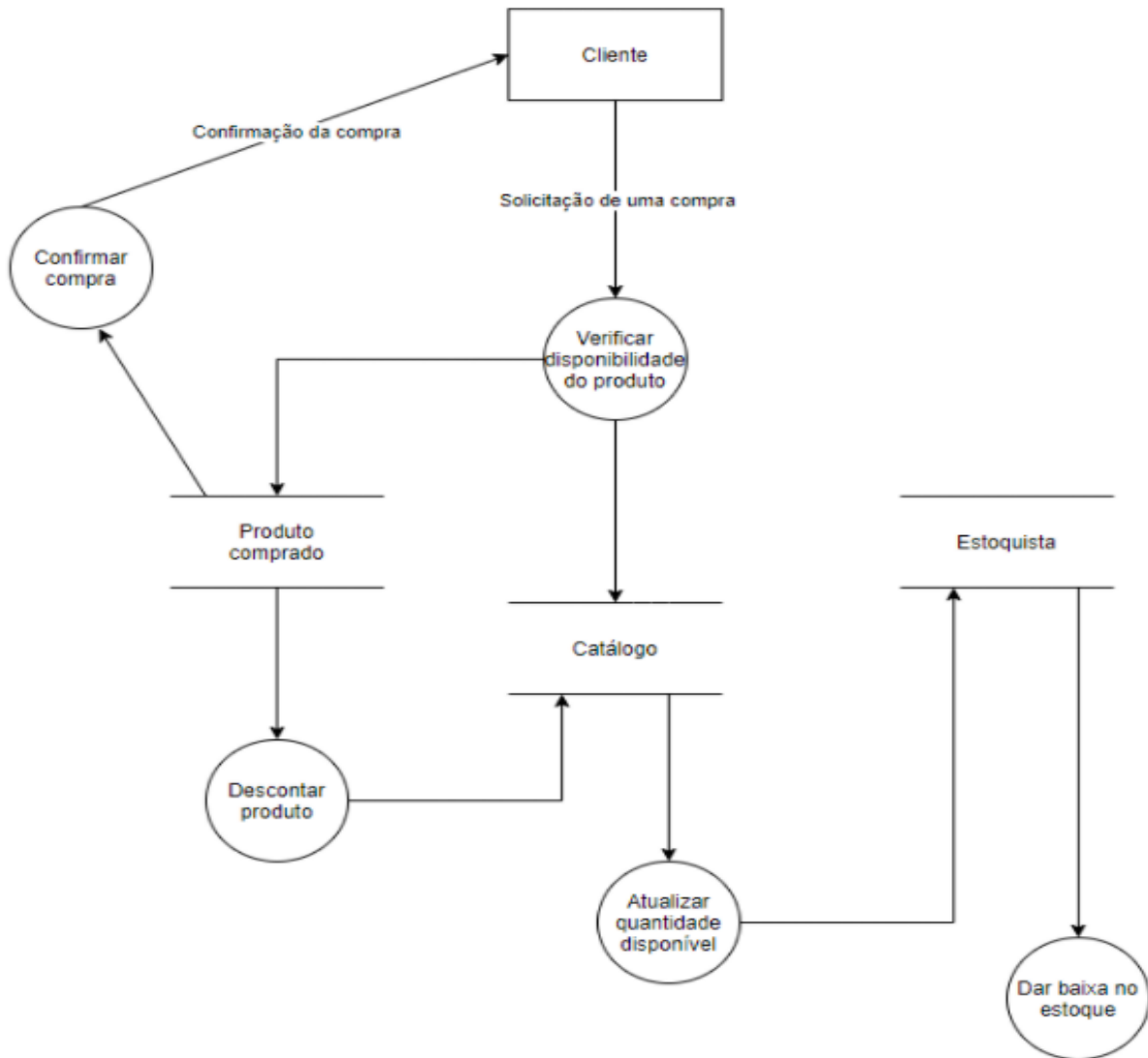
Fonte: Os autores.

2. Solução proposta

A solução proposta foi criar uma aplicação *web* que satisfizesse as necessidades do cliente. Foi desenvolvido um desenho de um diagrama de fluxo de dados que representasse a interação do usuário com o catálogo de produtos. O fluxo começa com um usuário da aplicação solicitando a compra de um item da loja virtual. Após o sistema confirmar que o produto se encontra disponível para aquisição, a compra é confirmada e o número de unidades compradas são automaticamente descontadas do catálogo, atualizando a quantidade ainda disponível.

Depois que os produtos foram descontados no sistema, é a vez do estoquista dar baixa no estoque e reservar para entrega os produtos comprados.

Figura 2 – DFD do catálogo de produtos.



Fonte: Os autores.

2.1. Requisitos do sistema

Tabela 2 – Requisitos.

| Requisito | Descrição |
|-----------|---|
| 0001 | O sistema deve ser autenticado. |
| 0002 | O sistema deve diferenciar os perfis de usuário. |
| 0003 | O sistema deve registrar dados iniciais dos clientes. |
| 0004 | O sistema deve apresentar o catálogo de produtos para os clientes. |
| 0005 | O cliente pode selecionar produtos em seu carrinho. |
| 0006 | O sistema deve informar a lista de produtos vinculado a um carrinho de cliente. |
| 0007 | O cliente pode informar seu endereço para entrega de produtos. |
| 0008 | O sistema deve realizar as transações das compras pela conta “Mercado Pago” das Sneakers. |
| 0009 | O sistema deve apresentar um resumo do pedido antes da finalização da venda. |
| 0010 | As informações do carrinho do cliente devem ser salvas a nível de sessão de usuário, caso o cliente não termine a compra. |

| | |
|------|--|
| 0011 | O sistema deve oferecer informações sobre os pedidos dos clientes. |
| 0012 | O cliente pode realizar alterar suas informações de cadastro. |
| 0013 | O administrador pode controlar o estoque da Sneakers. |
| 0014 | O administrador pode visualizar, ordenar e filtrar as vendas. |
| 0015 | O sistema deve ser seguro em relação aos dados transacionais. |

Fonte: Os autores.

2.2. Regras de negócio

Tabela 3 – Regras de negócio.

| Requisito | Descrição |
|-----------|--|
| 0001 | A Sneakers deve ter um sistema com divisões de perfis. |
| 0002 | O catálogo virtual deve ser controlado por um administrador. |
| 0003 | A Sneakers centraliza seus produtos de calçado por categorias pré-definidas. |
| 0004 | Os pagamentos recebidos devem ser via “Mercado Pago”. |
| 0005 | Os pedidos realizados devem conter status até a entrega. |
| 0006 | O estoque do catálogo deve refletir o estoque físico. |
| 0007 | Os produtos comprados devem dar baixa automaticamente no estoque. |

Fonte: Os autores.

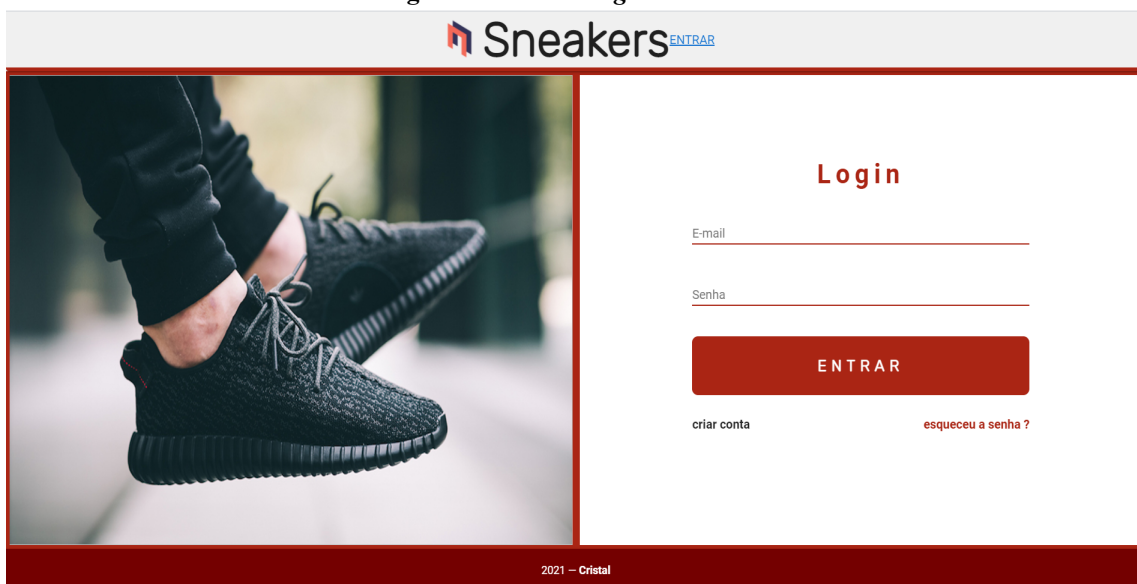
2.3. Descrição das funcionalidades

Dentre as funcionalidades do sistema, existem os alicerces de autenticação de usuário em uma aplicação *web*, sendo eles cadastro e *login*. Uma vez logado, o usuário poderá desfrutar dos serviços de compra de calçados que oferecemos. A aplicação também oferece suporte aos administradores e estoquistas em suas tarefas de manutenção de produtos, através de uma interface intuitiva e fácil de usar.

2.3.1. Login do usuário

O usuário deve informar seu *e-mail*, senha e clicar no botão “Entrar”. Caso o usuário tenha um perfil de “Cliente” ele será redirecionado para sua tela de catálogo. Caso ele seja um administrador, será redirecionado ao painel de administração. E caso não sejam válidas suas credenciais, a tela deve alertá-lo.

Figura 3 – Tela de *login* do cliente.



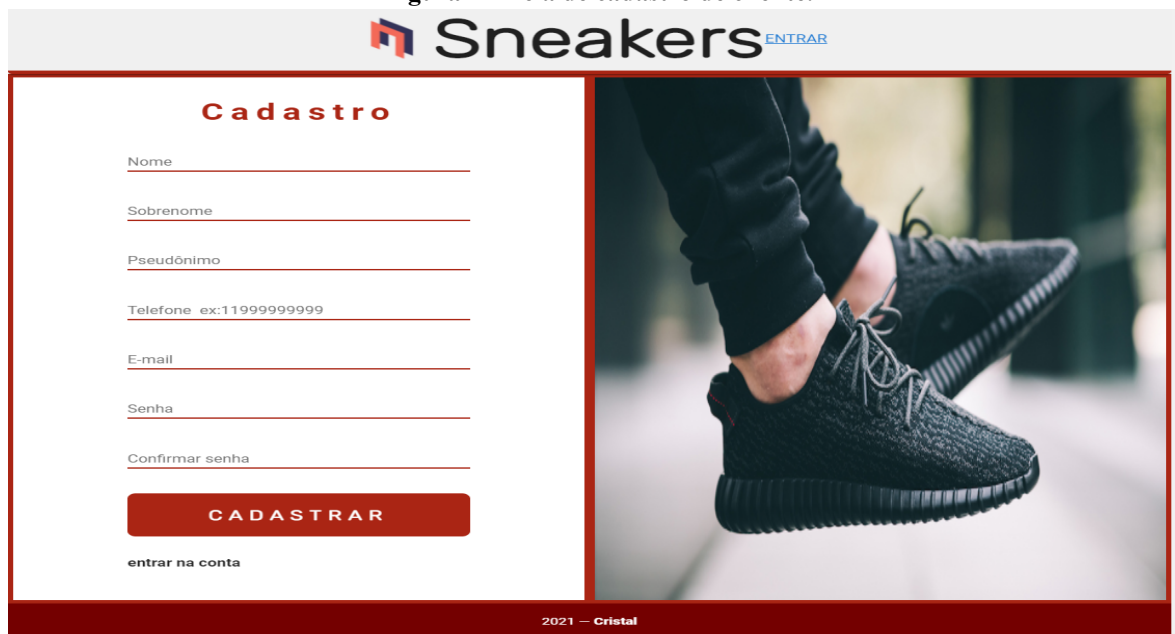
A interface de login do cliente para o site 'Sneakers'. No topo, há o logo 'Sneakers' com um ícone de sapato e o link 'ENTRAR' em azul. À esquerda, uma imagem de uma pessoa usando tênis pretos. À direita, o formulário de login com o título 'Login' em vermelho. O formulário contém campos para 'E-mail' e 'Senha', um botão vermelho 'ENTRAR', e links para 'criar conta' e 'esqueceu a senha?'. O rodapé indica '2021 - Cristal'.

Fonte: Os autores.

2.3.2. Cadastro de cliente

É nesta tela que o cliente irá cadastrar sua conta, adicionando em todos os campos suas informações pessoais, como nome, sobrenome, pseudônimo, telefone, *e-mail* e sua senha desejada. A senha deve conter no mínimo oito caracteres e deve ser confirmada depois. Caso os pré-requisitos para cadastrar a senha não sejam satisfeitos, assim como houver um campo vazio ou tiver alguma informação inválida, o cadastro não é efetivado. Após um cadastro bem sucedido, o usuário é redirecionado para a tela de *login*.

Figura 4 – Tela de cadastro de cliente.



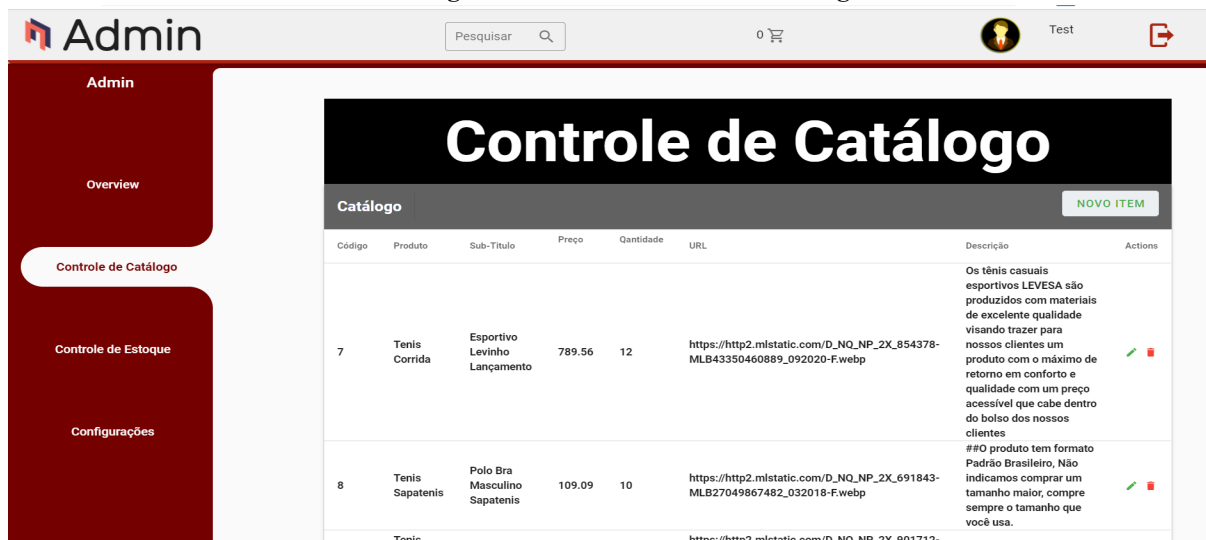
A interface de cadastro de cliente para o site 'Sneakers'. No topo, há o logo 'Sneakers' com um ícone de sapato e o link 'ENTRAR' em azul. À esquerda, o formulário de cadastro com o título 'Cadastro' em vermelho. O formulário contém campos para 'Nome', 'Sobrenome', 'Pseudônimo', 'Telefone ex:11999999999', 'E-mail', 'Senha' e 'Confirmar senha', um botão vermelho 'CADASTRAR', e um link para 'entrar na conta'. À direita, uma imagem de uma pessoa usando tênis pretos. O rodapé indica '2021 - Cristal'.





Fonte: Os autores.

2.3.3. Controle de catálogo

Esta é a interface onde as alterações de produtos são efetivadas. Apenas o administrador e o estoquista possuem permissão para acessá-la e realizar mudanças quando julgarem necessárias. Todas as alterações refletem diretamente no catálogo visualizado pelo administrador, sejam elas os títulos dos produtos, subtítulos, preços, descrições, quantidade disponível, imagem e número de identificação do produto.

Figura 5 – Tela do controle de catálogo.



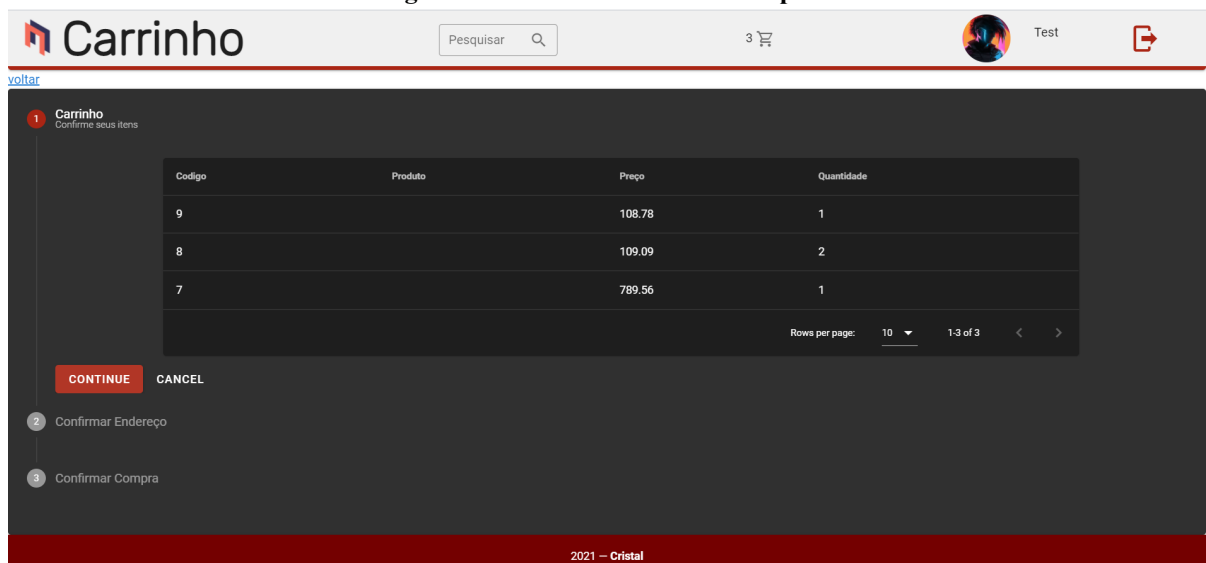
| Código | Produto | Sub-Título | Preço | Quantidade | URL | Descrição | Actions |
|--------|-----------------|------------------------------|--------|------------|---|---|---|
| 7 | Tenis Corrida | Esportivo Levinho Lançamento | 789.56 | 12 | https://http2.mlstatic.com/D_NQ_NP_2X_854378-MLB43350460889_092020-F.webp | Os tênis casuais esportivos LEVESA são produzidos com materiais de excelente qualidade visando trazer para nossos clientes um produto com o máximo de retorno em conforto e qualidade com um preço acessível que cabe dentro do bolso dos nossos clientes |   |
| 8 | Tenis Sapatenis | Polo Bra Masculino Sapatenis | 109.09 | 10 | https://http2.mlstatic.com/D_NQ_NP_2X_691843-MLB27049867482_032018-F.webp | ##O produto tem formato Padrão Brasileiro, Não indicamos comprar um tamanho maior, compre sempre o tamanho que você usa. |   |

Fonte: Os autores.

2.3.4. Carrinho de compras

Ao entrar na tela do carrinho, o cliente pode ver e alterar detalhes de sua compra, como quantidade de produtos e produtos distintos que deseja comprar. Também deve informar o endereço de entrega e confirmar a compra.

Figura 6 – Tela do carrinho de compras.



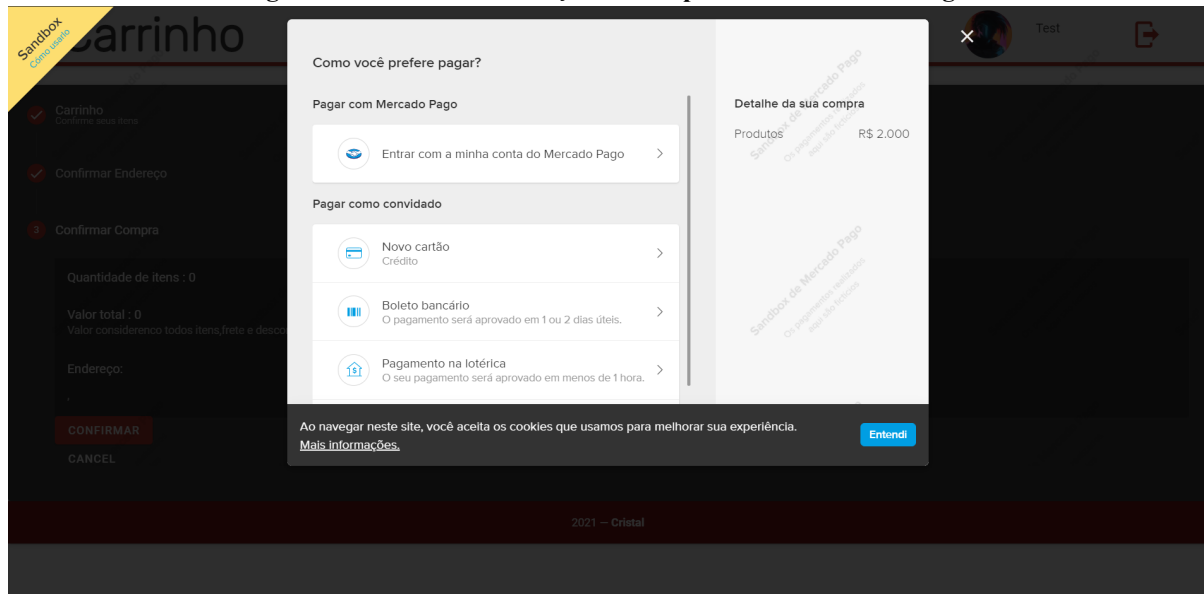
| Código | Produto | Preço | Quantidade |
|--------|---------|--------|------------|
| 9 | | 108.78 | 1 |
| 8 | | 109.09 | 2 |
| 7 | | 789.56 | 1 |

Fonte: Os autores.

2.3.5. Conclusão da venda – Mercado Pago

Após completar o pedido, uma tela do mercado pago aparece, onde o cliente deverá adicionar os dados da forma de pagamento escolhida. O pedido é então processado e os dados de compra do cliente são protegidos pelo pagamento seguro do Mercado Pago.

Figura 7 – Tela de confirmação de compra com o Mercado Pago.



Fonte: Os autores.

3. Projeto, análise e implementação

O sistema consiste em um site em que o frontend foi construído em Vue.js e o *back-end* monolítico construído com NodeJS. O *back-end* provê uma API com a maior parte das funcionalidades necessárias para o funcionamento dos fluxos no site. Adicionalmente temos apenas uma integração externa para processamento de pagamentos que é feita pela API do Mercado Pago.

O banco de dados escolhido foi o MySQL, um banco de dados relacional que atende bem a proposta de solução dada o escopo desse projeto e os modelos envolvidos.

A arquitetura segue um modelo convencional de cliente-servidor, tendo as interfaces providas ao usuário pelo navegador e comunicando com o servidor através do protocolo HTTP.

Tanto o servidor *back-end* quanto o código *front-end* estão hospedados na plataforma Heroku com o banco de dados na RDS AWS.

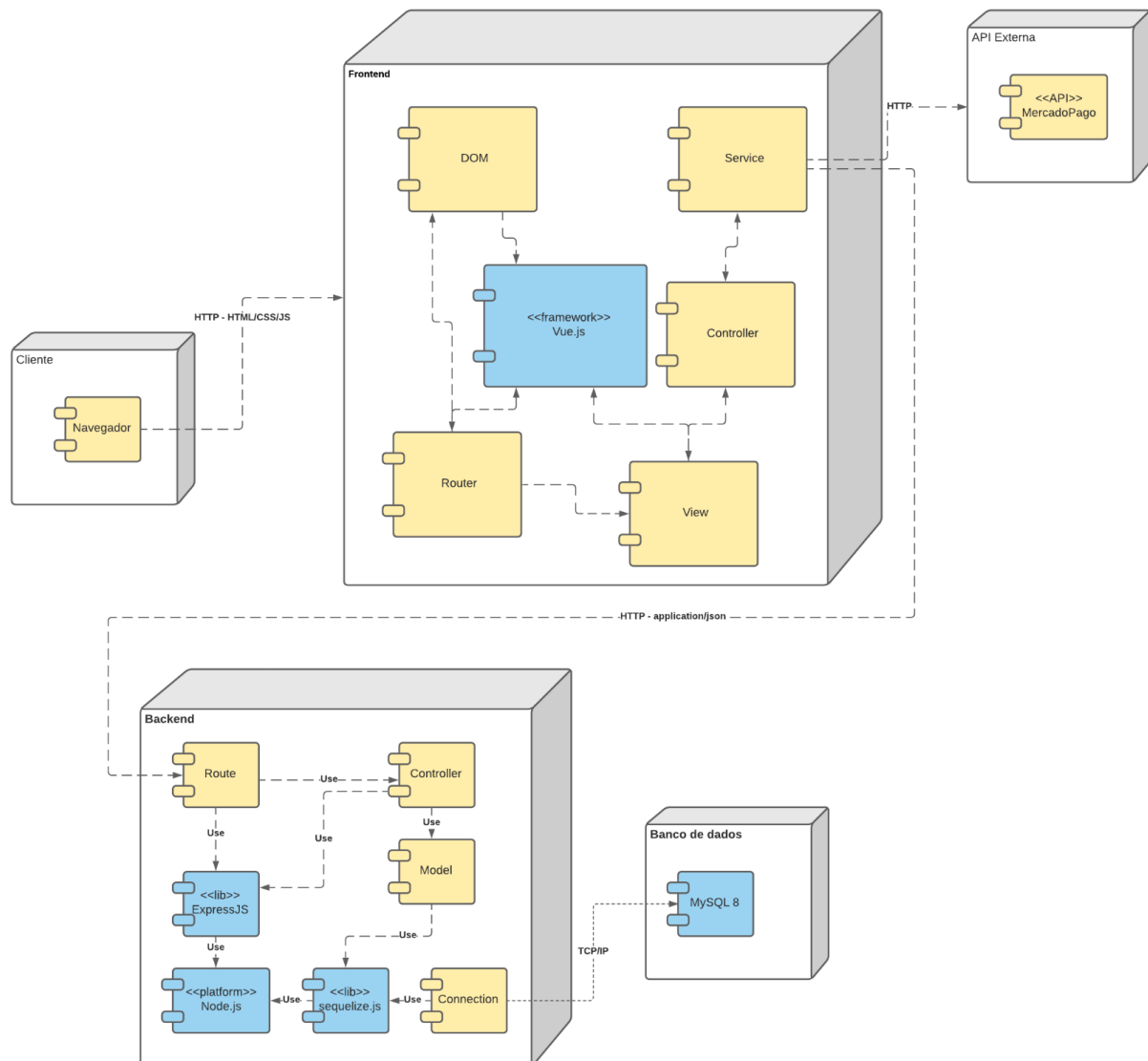
3.1. Arquitetura, módulos e subsistemas

Seguindo o modelo cliente-servidor, toda a parte que tange ao lado “cliente” é responsabilidade do módulo de *front-end*: a resolução de URIs do navegador para renderizar as telas e buscar informações necessárias no servidor assim como enviar informações de acordo com as interações do usuário com os fluxos disponíveis nas telas. E todo o resto, no que tange o lado “servidor”, é responsabilidade do módulo de *back-end*, provendo uma API que possibilita operações elementares de CRUD sobre as entidades do banco de dados.

O provisionamento da arquitetura consiste essencialmente em 3 partes distintas, que na nossa proposta é representado por 3 diferentes plataformas cloud que fornecem serviços PaaS com a infraestrutura necessária para hospedarmos as partes do sistema: uma para hospedar o

código estático do *front-end* e servir em uma DNS, outro para hospedar o código do *back-end* rodando e respondendo às requisições do cliente e por fim, o banco de dados que é hospedado separadamente em uma plataforma especializada em provisionamento de bancos de dados.

Figura 8 – Diagrama de implantação.



Fonte: Os autores.

3.1.1. Módulo *back-end*

Consiste em um serviço único, responsável por prover API para todas entidades e funcionalidades do sistema, desde autenticação às operações CRUD sobre as entidades.

3.1.2. Módulo *front-end*

Módulo responsável pelo gerenciamento de estado, renderização e estilização das telas renderizadas no navegador do cliente, juntamente com a integração com a API do *back-end*.

3.2. Projeto de banco de dados

Tabela 4 – Detalhamento das tabelas e campos do banco de dados.

| Tabela: produto | | |
|---|---|---|
| Contém informações cadastrais de cada produto da loja, principalmente utilizadas na listagem do produto na tela de catálogo de produtos, juntamente com o preço atual de venda utilizado no cálculo do valor total dos pedidos. | | |
| Campo | Tipo | Descrição |
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| nome | varchar(255) | Nome do produto. |
| subtitulo | varchar(255) | Informações adicionais utilizado como complemento ao nome do produto. |
| descricao | text | Descrição detalhada do produto. |
| precoAtual | decimal(8,2) | Preço de venda atual do produto. |
| urlImage | varchar(2048) | URL da imagem do produto exibido no catálogo. |
| categoria | enum('Corrida', 'Skateboarding', 'Academia', 'Sportwear', 'Acessórios') | Lista de possíveis categorias a que um produto pode pertencer. |
| quantidade | bigint | Determina a disponibilidade do produto em estoque. |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |
| Tabela: item_carrinho | | |
| Representa cada produto selecionado pelo cliente para ser incluído no carrinho. Cada entrada nessa tabela contém também a quantidade de unidades de cada produto adicionado no carrinho e o preço daquele produto naquele momento, que será utilizado para cálculo do valor total posteriormente. | | |
| Campo | Tipo | Descrição |
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| idCarrinho | bigint (FK) | Id único do carrinho. <i>Foreign key</i> para a tabela carrinho. Serve essencialmente para fazer a ligação de um item dentro de determinada entidade de carrinho. |

| | | |
|------------|--------------|---|
| idProduto | bigint (FK) | Id único do produto. <i>Foreign key</i> para a tabela produto. Serve essencialmente para identificar qual o produto que está sendo adicionado a determinado carrinho. |
| quantidade | int | Determina a quantidade de unidades do mesmo produto inserido dentro de um carrinho. |
| precoVenda | decimal(8,2) | Preço de venda do produto no momento da inserção do produto dentro do carrinho. Diferente do campo “precoAtual” dentro da tabela produto, esse é um valor de produto que pode ser diferente dependendo do momento da compra, onde a fonte da verdade do preço atual sempre será o que está salvo na tabela produto. |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |

Tabela: carrinho

Representa a entidade que agrupa os itens de interesse de um cliente e o endereço de entrega. Possui seu próprio ciclo de vida controlado pelo campo “status” e serve de insumo para a continuidade do ciclo de vida da entidade “pedido”.

| Campo | Tipo | Descrição |
|--------------|--|---|
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| idCliente | bigint (FK) | Id do cliente. <i>Foreign key</i> para a tabela “cliente”. Identifica a qual cliente pertence determinado carrinho. |
| idEndereco | bigint (FK) | Id do endereço. <i>Foreign key</i> para a tabela “endereco”. No momento do <i>checkout</i> , ao selecionar o endereço de entrega do pedido é salvo ao carrinho a referência do endereço no carrinho. |
| status | enum(‘OPEN’, ‘SUCCESS’, ‘CANCELED’, ‘EXPIRED’) | Uma lista de estados possíveis para um carrinho, representando seu ciclo de vida. “OPEN” - representa o estado inicial do carrinho ao ser criado. Enquanto não for efetivado a compra para esse registro de carrinho, ele ficará nesse estado. “SUCCESS” - representa a transição de estado depois que um pedido nasce a partir desse carrinho. “CANCELED” - representa o cancelamento de determinado pedido que pode ser feito pela |

| | | administração da loja. “EXPIRED” - representa a invalidação automática do carrinho por estar muito tempo aberto. |
|--|---|--|
| desconto | decimal(8,2) | Valor monetário dedutível do preço total do carrinho. |
| dataExpiracao | datetime | Data limite para efetivação da compra de determinado carrinho. Caso exceda o tempo pré-calculado, o <i>status</i> do carrinho é automaticamente atualizado para “EXPIRED”. |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |
| Tabela: pedido | | |
| Um pedido sempre nasce quando um “carrinho” é fechado, ou uma compra é efetivada a partir de um carrinho. Todas informações de produto, valor total e endereço de entrega são atribuídos ao “carrinho” durante o ciclo de vida dele, o pedido contém apenas uma referência ao “carrinho” de onde se busca essas informações. O papel principal da tabela “pedido” é controlar o ciclo de vida após a confirmação de um pedido de compra. | | |
| Campo | Tipo | Descrição |
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| idCliente | bigint (FK) | Id do cliente. <i>Foreign key</i> para a tabela “cliente”. Identifica a qual cliente pertence determinado pedido. |
| idCarrinho | bigint (FK) | Id do cliente. <i>Foreign key</i> para a tabela “carrinho”. Identifica o carrinho que originou esse pedido de compra. |
| status | enum(‘COMPLETED’, ‘CANCELED’, ‘AWAITING_PAYMENT’, ‘CONFIRMED’, ‘DISPATCHED’, ‘IN_TRANSIT’) | Uma lista de estados possíveis para um pedido, representando seu ciclo de vida. “AWAITING_PAYMENT” - estado inicial do pedido logo após confirmação do pedido. Até que haja uma atualização imperativa para o próximo estado, ele se manterá. “CONFIRMED” - representa a confirmação de que a loja possui todas as informações e pagamentos necessários para proceder com o envio do pedido. “CANCELED” - representa o cancelamento do pedido. “DISPATCHED” - representa a passagem do pedido para a transportadora. |

| | | |
|-----------|----------|---|
| | | <p>“IN_TRANSIT” - representa a sinalização da transportadora de quando a entrega está na última etapa.</p> <p>“COMPLETED” - representa a completude do pedido, quando ele é entregue com sucesso.</p> |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |

Tabela: endereco

Armazena os endereços de cada cliente. É utilizado durante o *checkout* para determinar o endereço de entrega do pedido.

| Nome do campo | Tipo | Descrição |
|---------------|-------------|---|
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| idCliente | bigint (FK) | Id do cliente. <i>Foreign key</i> para a tabela “cliente”. Identifica a qual cliente pertence determinado endereço. |
| uf | varchar(3) | Sigla do estado do endereço (ex.: SP, RJ). |
| cidade | varchar(50) | Cidade do endereço. |
| bairro | varchar(50) | Bairro do endereço. |
| rua | varchar(50) | Rua do endereço. |
| numero | varchar(10) | Número do endereço (ex.: 35, 47B, 2C-3). |
| complemento | varchar(30) | Complemento do endereço (ex.: apartamento 10, casa 8). |
| cep | varchar(10) | CEP do endereço. |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |

Tabela: cliente

Armazena os dados cadastrais e de autenticação do cliente e dos administradores do sistema.

| Nome do campo | Tipo | Descrição |
|---------------|---------------------------|---|
| id | bigint (PK) | Id único e <i>primary key</i> da tabela. |
| nome | varchar(255) | Primeiro nome do cliente. |
| sobrenome | varchar(255) | Sobrenome do cliente. |
| pseudonimo | varchar(255) | <i>Nickname</i> a ser utilizado nas telas para se referir ao cliente. |
| telefone | varchar(255) | Telefone de contato do cliente. |
| email | varchar(500) | <i>E-mail</i> do cliente. É utilizado para se autenticar no sistema. |
| senha | varchar(255) | Senha do cliente. É utilizado para se autenticar no sistema. |
| senhaSalt | varchar(255) | <i>Hash</i> utilizado para recuperação de senha. |
| profile | enum('ADMIN', 'CUSTOMER') | Perfil de acesso do cliente. “ADMIN” - administradores “CUSTOMER” - cliente |
| imgProfile | varchar(500) | URL da foto de perfil do cliente. |
| createdAt | datetime | Campo auto-gerado para determinar a data de gravação de um produto na tabela. |
| updatedAt | datetime | Campo auto-gerado para determinar a data de atualização mais recente de um produto na tabela. |

Fonte: Os autores.

3.3. Estrutura do sistema

Em termos de como a tabela é utilizada nos principais fluxos do sistema, para a autenticação e identificação do cliente dentro do sistema, é utilizada apenas a tabela “cliente”. No momento em que o cliente se autentica e acessa a plataforma, uma nova linha é criada na tabela “carrinho” para que já fique disponível um carrinho aberto.

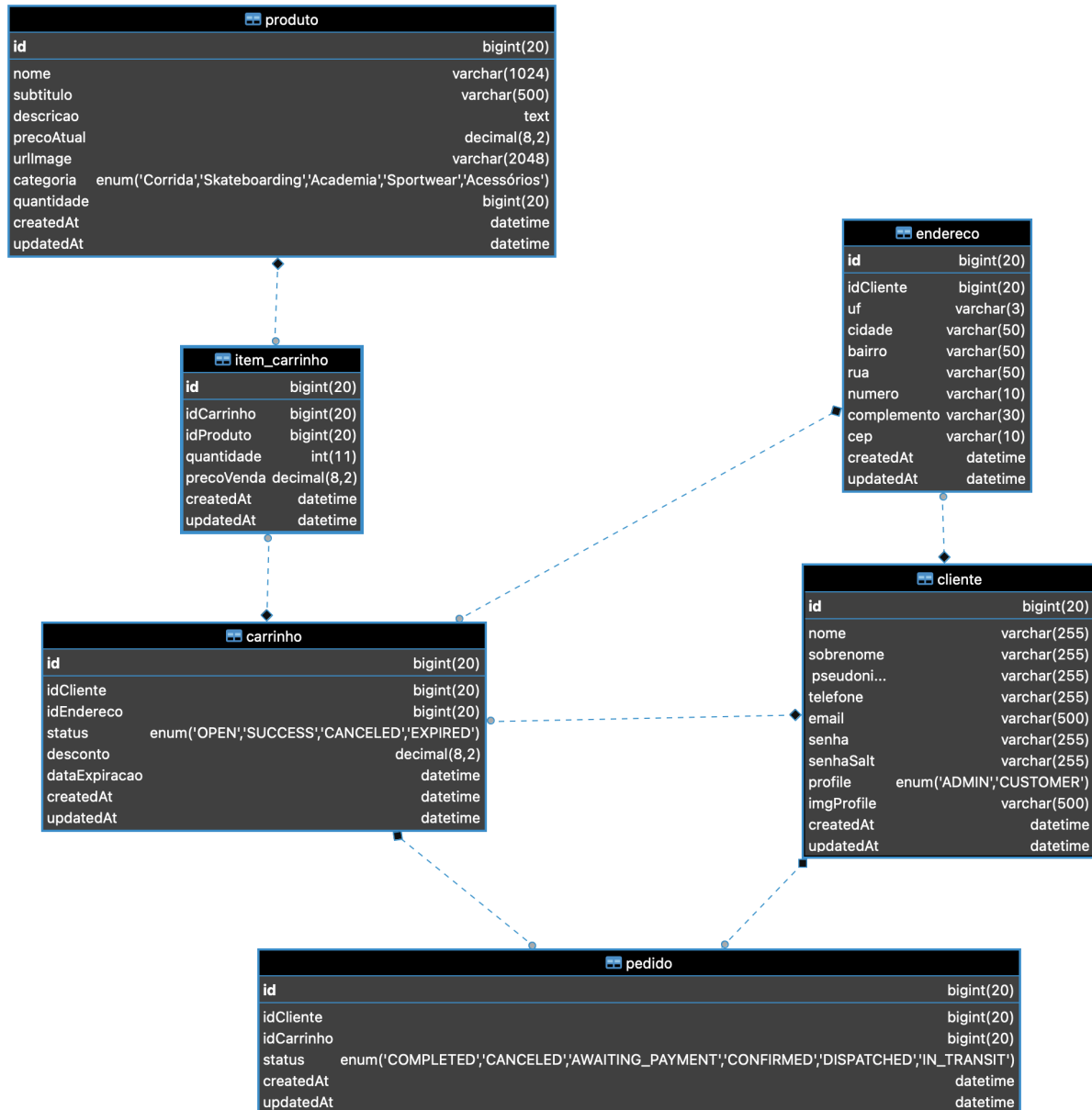
Já na tela de listagem de produtos consultados a tabela “produto” para buscar informações para exibição dos detalhes de cada item listado, esses mesmos dados são salvos no LocalStorage do navegador quando adicionado ao carrinho, além de enviar uma requisição de criação de um novo “item-carrinho” para o servidor salvar no banco, isso possibilita ter o estado atual do carrinho persistido no servidor, e caso o cliente acesse a conta de outro local será possível visualizar tudo que foi feito inicialmente em outro dispositivo.

Na tela de checkout é utilizado a referência do cliente logado para buscar o carrinho e os itens associados a determinado carrinho, no passo seguinte quando o endereço de entrega é selecionado, a referência do mesmo é vinculado ao carrinho aberto, e por fim quando a compra é finalizada, o estado do carrinho para é atualizado para “SUCCESS” e uma nova entrada de é criada na tabela “pedido” com o estado inicial “AWAITING_PAYMENT”.

Somente após o cliente finalizar o fluxo de pagamento no Mercado Pago e o sistema receber uma confirmação do pagamento, atualizamos o estado do mesmo pedido para “CONFIRMED”, todos os estados posteriores a esse é controlado pela administração da loja.

A figura 9 representa o diagrama do banco de dados mostrando a modelagem e relações das tabelas do sistema.

Figura 9 – Diagrama do banco de dados.



Fonte: Os autores.

A estrutura do código seguiu como um “mono-repo” sendo divididas pelas pastas “backend” e “frontend”, existe também na raiz da estrutura de pastas uma pasta “resource” que centraliza alguns artefatos de natureza mais documental que são compartilhados entre ambos contextos.

A organização lógica do código segue uma abstração do MVC tanto no *back-end* quanto no *front-end*. A separação macro entre as pastas “backend” e “frontend” representa bem como os serviços são provisionados de forma independente, dentro de cada uma dessas pastas existe uma série de *scripts* que é responsável por subir ou compilar o código para a versão de

distribuição, entre outras especificidades de cada contexto que é isolado por essa separação de pastas.

A figura 8 (diagrama de implantação) representa bem como cada bloco lógico de código interage internamente.

3.4. Tecnologias utilizadas

Tabela 5 – Tecnologias, ferramentas e plataformas utilizadas.

| Tecnologia | Justificativa |
|-------------------|---|
| Git | Ferramenta para execução de comandos de versionamento de código. |
| GitHub | Plataforma onde o código-fonte é hospedado e versionado. URL do projeto: https://github.com/ope-cristal/OPE-Cristal . |
| Heroku | Utilizado para hospedar o <i>back-end</i> e <i>front-end</i> . |
| RDS AWS | Serviço PaaS da AWS utilizado para hospedagem do banco de dados. |
| Trello | Plataforma utilizada para gerenciamento das tarefas do projeto. |
| LucidChart | Plataforma utilizada para criação dos diagramas do projeto. |
| G Suite | Conjunto de ferramentas da Google utilizadas para escrita das documentações (Google Docs) e armazenamento dos documentos (Google Drive). |
| MySQL | SGBD relacional para persistência dos dados relacionados a todas entidades do sistema. |
| MySQL Workbench | Ferramenta utilizada para visualização e interação com o banco de dados. |
| HTML | Linguagem de marcação padrão dos navegadores e utilizado para renderizar as páginas do <i>site</i> . |
| CSS | Código de estilização de páginas padrão dos navegadores utilizado para manipular a aparência e comportamentos dos elementos do <i>site</i> . |
| Javascript | Linguagem de programação responsável pela lógica dos componentes nas páginas do site, utilizada para desenvolver tanto o <i>front-end</i> quanto o <i>back-end</i> do projeto. |
| VueJS | Framework de Javascript para construção de interfaces utilizada no <i>front-end</i> . |
| Vuetify | Biblioteca utilizada para simplificar e padronizar a estilização do <i>front-end</i> . |
| Vue-router | Biblioteca responsável pelo controle de renderização dos componentes de interface com base na rota (URI) no <i>front-end</i> . |
| Vuex | Biblioteca responsável pelo gerenciamento de estado da aplicação no <i>front-end</i> . |

| | |
|--------------|--|
| Mercado Pago | SaaS do Mercado Livre utilizado para processamento de pagamentos. Acessado através de biblioteca do mesmo nome pelo <i>front-end</i> . |
| Axios | Biblioteca Javascript utilizada no <i>front-end</i> para abstração das requisições HTTP. |
| NodeJS | Plataforma de desenvolvimento utilizada na construção do <i>back-end</i> do projeto. |
| Sequelize | Biblioteca de ORM para comunicação com BD no <i>back-end</i> . |
| Express | Framework web para construção das APIs no <i>back-end</i> . |
| Prettier | Ferramenta de padronização de estilo de formatação de código. |
| Docker | Ferramenta de containerização utilizada para o provisionamento local do banco de dados para fins de desenvolvimento no <i>back-end</i> . |

Fonte: Os autores.

4. Considerações finais

A solução proposta foi bem aceita pelo cliente, principalmente por atender as necessidades do negócio de forma completa, sem complexidades adicionais em termos de usabilidade que tornou o uso da plataforma funcionalmente agradável.

O cliente possui uma carteira de clientes assíduos da época de atendimento presencial e espera abordá-los em breve para entender a experiência destes dentro da nova plataforma e entender se existe necessidade de um projeto futuro para implementar melhorias nesta versão inicial do *site*.

O progresso do desenvolvimento do projeto seguiu de forma pouco linear, principalmente pelo momento de pandemia em que o mundo vive que inevitavelmente impactou a produtividade de todo o time. De forma geral, o gerenciamento do projeto seguindo algumas das diretrizes do *scrum* facilitou o acompanhamento do progresso ao longo do desenvolvimento com os *boards* e reuniões periódicas para priorização das tarefas.

Dado todas as variáveis mencionadas e os recursos que o time tinha em mãos, o resultado da implementação no fim foi considerado satisfatório pela equipe.

Para visualizar a plataforma, acesse: <https://ope-cristal-frontend.herokuapp.com/>.

Agradecimentos

Agradecimentos à instituição de ensino Faculdade Impacta Tecnologia pela oportunidade de aplicar os conhecimentos teóricos adquiridos ao decorrer do curso, aos professores e professoras pela preparação e disponibilização dos materiais, aos coordenadores e coordenadoras das OPEs pelas orientações e acompanhamento do desenvolvimento do projeto.