UNIVERSIDADE ESTADUAL DA PARAÍBA

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

DEPARTAMENTO DE COMPUTAÇÃO - DC

ALYSSANDRO DYOGO PEREIRA RAMOS

JOSÉ LUCAS CUSTÓDIO SILVA

DOCUMENTO DE VISÃO

CAMPINA GRANDE - PB

2025

Sumário

● Descrição do Sistema

O sistema "Priscyla Store" é uma plataforma full-stack desenvolvida para gerenciar uma loja de roupas online. Ele permite a administração de produtos, usuários e pedidos, integrando um back-end desenvolvido em Node.js com um front-end construído em React e um banco de dados MongoDB.

● Requisitos Funcionais

- **Controle de Usuários:** O usuário deve ser capaz de efetuar cadastro e login na plataforma.
- **Compras de Produtos:** O sistema deve permitir os usuários a realizar compras de produtos na plataforma.
- **Gerência de Produtos Disponíveis:** O administrador deve ser capaz de gerenciar o controle de produtos disponíveis aos usuários para compra, ou seja, cadastro de novos produtos e reposição dos já existentes.
- **Acesso a Informações de Compras:** O administrador deve ter acesso aos pedidos de compra com todas as informações necessárias para efetuar o envio ao comprador.

● Requisitos Não Funcionais

- **Responsividade:** O sistema deve ser responsivo a plataformas desktop e mobile.
- **Criptografia de Dados de Usuário:** O sistema deve garantir a segurança dos dados dos usuários com base no **Bcrypt**.
- **Interface Intuitiva:** O sistema deve seguir boas práticas de **UX/UI**, garantindo navegação intuitiva.
- **API Protegida:** O acesso à API deve ser protegido via JWT (Json Web Token).
- **Ato de Compra Simples:** O processo de compra não deve ter mais de **8 passos** para evitar desistências.

- **Interface Compatível:** O front-end deve ser compatível com os navegadores mais populares (**Chrome, Firefox, Edge**).

- **Métodos de Pagamentos:** O sistema deve permitir integração com **métodos de pagamento** como **Pix, cartão de crédito e boleto**.

- **Confirmação de Pagamento:** O sistema deve ser capaz de atualizar os dados de compras realizadas após confirmação de pagamento através da API.

- **Banco de Dados NoSQL:** O back-end deve ser compatível com o banco de dados NoSQL **MongoDB.**

● Perfis de Usuários

1. Administrador

- **Descrição:** Usuário responsável por gerenciar a loja, incluindo produtos, usuários e pedidos.
- **Tipo:** Interno (Gerente da loja)
- **Responsabilidades:**
  - Cadastrar, editar e excluir produtos
  - Gerenciar contas de clientes
  - Monitorar pedidos e status de vendas
  - Administrar o estoque e atualizar informações sobre os produtos.

2. Cliente

- **Descrição:** Usuário que acessa a plataforma para visualizar e comprar produtos.
- **Responsabilidades:**
  - Criar e gerenciar conta.
  - Navegar pelos produtos e realizar compras.
  - Acompanhar status de pedidos e entregas.
  - Gerenciar informações de pagamento e endereço de entrega.

● Tecnologias e Ferramentas

1. Front-End

- **HTML:** Utilizada para marcação dos elementos gráficos na interface de usuários
- **CSS:** Permite a estilização dos elementos gráficos aos usuários a fim de melhorar a experiência deles com a aplicação
- **Javascript:** Usada para aplicar uma maior dinamicidade à interface dos usuários e permitir uso da biblioteca React, além de também permitir a comunicação entre a interface e a API da aplicação, dentre outras.
- **React:** Permite a componentização da aplicação de modo a desenvolver por partes menores e nos dá a possibilidade de uso de componentes prontos ou reutilização de outros já criados.
- **Bootstrap e React-Bootstrap:** Fornece ícones, imagens, componentes e estilizações já prontas apenas para aplicar de forma mais fácil e acelerando assim o desenvolvimento e uma melhor experiência aos usuários.

2. Back-End

- **NodeJS:** Usado para desenvolver o back-end da aplicação de forma simples e prática. Também foi usado para administrar bibliotecas e demais frameworks no mesmo.
- **Node Express:** Usado para acelerar o desenvolvimento do back-end de forma a termos recursos mínimos para desenvolvê-lo sem maiores problemas.
- **Bcrypt:** Usado para criptografia dos dados sensíveis dos usuários no ato de cadastrar seus dados e grava-los no banco de dados, assim como também qualquer outro dado de maior sigilo.
- **JWT:** Usado para emitir um token de autorização ao usuário após. Esse token permite que a aplicação siga a realizar suas atividades sem necessidade constante de consulta dos dados de usuário ao banco, apenas usando esse token resguardado temporariamente LocalStorage do navegador. Esse token terá validade até o usuário deslogar da aplicação.
- **CORS:** O Cors será usado para permitir requisições através de um mesmo domínio, à princípio ele deve estar presente na aplicação tanto para testes durante o desenvolvimento quanto posteriormente caso seja necessário.

- **dotenv:** A dotenv será usada para capturar e utilizar as variáveis locais da aplicação, podendo ser essas variáveis desde a senha do banco de dados até as portas específicas de cada parte da aplicação durante o desenvolvimento.
- **Mongoose:** O mongoose será o framework que fará a comunicação entre o back-end da aplicação com o banco de dados.
- **Multer:** O Multer é um Middleware que permitirá o upload de dados ao servidor da aplicação, mais especificamente das imagens de produtos cadastrados.
- **Docker**: Ferramenta para conteinerização da aplicação, garantindo portabilidade e facilidade de implantação.

Conclusão

Este documento define a visão geral do sistema **Priscyla Store**, abrangendo suas principais funcionalidades, requisitos e tecnologias utilizadas. O objetivo é garantir uma experiência eficiente para administradores e clientes, proporcionando uma plataforma robusta e intuitiva para a gestão e compra de produtos.