A seguir, um documento detalhado que descreve como o projeto "webmarket" está estruturado e funcionando. Este resumo abrange tanto o backend (NestJS com Prisma e PostgreSQL) quanto o frontend (Next.js com Tailwind CSS e NextAuth), além de descrever os fluxos de dados, as funcionalidades implementadas e a organização geral dos arquivos.

**Resumo Detalhado do Projeto WebMarket**

**1. Visão Geral**

**Objetivo:**  
O projeto "webmarket" é um sistema de e-commerce que oferece:

* Listagem e filtragem de produtos com paginação.
* Exibição de produtos destacados na home page.
* Páginas de detalhe do produto (com opção de adicionar ao carrinho).
* Autenticação de usuários via NextAuth.
* Gerenciamento do carrinho e checkout.
* Área para administração/inventário e perfil do usuário.

**Principais Tecnologias:**

* **Frontend:**
  + **Next.js** – Framework React para SSR e SSG.
  + **Tailwind CSS** – Utilitário para estilização rápida e responsiva.
  + **NextAuth.js** – Gerenciamento de autenticação.
  + **Context API** – Para gerenciamento do carrinho (CartContext).
* **Backend:**
  + **NestJS** – Framework Node.js para construir APIs escaláveis e bem estruturadas.
  + **Prisma** – ORM para acesso ao banco de dados PostgreSQL.
  + **ServeStaticModule** – Serve arquivos estáticos (por exemplo, imagens) do diretório do projeto.
* **Banco de Dados:**
  + **PostgreSQL** – Utilizado para armazenar os dados dos produtos e demais informações do sistema.

**2. Estrutura de Arquivos e Organização**

**Backend (webmarket/backend)**

* **/dist:**
  + Diretório gerado após o build com os arquivos compilados em JavaScript.
* **/imagestest2:**
  + Contém as imagens dos produtos. Exemplo:
    - C00049/image\_1.jpg
    - C00050/image\_1.jpg
* **/node\_modules:**
  + Dependências instaladas do Node.js.
* **/prisma:**
  + Arquivo schema.prisma define o modelo do banco de dados.
  + Exemplo de modelo: a tabela produtos possui campos como product\_code (chave primária), name, preco, category, subcategory\_level\_1, etc.
* **/products:**
  + **products.controller.ts:** Define os endpoints HTTP para consultar produtos e filtros.
  + **products.service.ts:** Contém a lógica de negócio para aplicar filtros, paginação e realizar consultas no banco via Prisma.
  + **products.module.ts:** Conecta o controller e o service.
  + **Arquivos de teste (spec):** Para garantir a funcionalidade dos módulos de produtos.
* **/src:**
  + **app.module.ts:** Configura o aplicativo globalmente, importando o módulo de produtos e configurando o ServeStaticModule para servir imagens (configurado para usar o diretório imagestest2).
  + **app.controller.ts, app.service.ts:** Possíveis endpoints e lógica global.
  + **main.ts:** Inicializa o NestJS, habilita CORS, define o prefixo global "api" (todos os endpoints ficam sob /api) e inicia o servidor na porta 4000.
  + **prisma.service.ts:** Serviço para conectar o Prisma ao NestJS (inclui métodos onModuleInit e onModuleDestroy).
* **.env:**
  + Contém a variável DATABASE\_URL que conecta ao PostgreSQL.

**Frontend (webmarket/frontend)**

* **.env.local:**
  + Contém variáveis de ambiente para desenvolvimento, como NEXT\_PUBLIC\_API\_URL (que aponta para o backend) e NEXTAUTH\_URL.
* **/public:**
  + Arquivos públicos (imagens, ícones, etc.) utilizados pelo Next.js.
* **/src/app:**
  + **page.tsx:** A home page, que exibe um banner (Hero Section) com data atual, produtos destacados e categorias.
  + **/products/page.tsx:** Página de listagem de produtos com painel lateral para filtros (categoria, subcategoria, marca, disponibilidade e faixa de preço) e paginação.
  + **/products/[product\_code]/page.tsx:** Página de detalhe do produto, onde o usuário pode ver informações detalhadas e adicionar ao carrinho.
  + **Outras páginas:** Para autenticação (login), perfil, carrinho e checkout.
* **/src/components:**
  + **ProductCard.tsx:** Componente reutilizável para exibir informações dos produtos – usado em múltiplas páginas. Inclui botão “Adicionar ao Carrinho” e, possivelmente, um controle para ajustar a quantidade.
  + **Navbar.tsx, Footer.tsx, Hero.tsx, Categories.tsx, FeaturedProducts.tsx:** Componentes de interface que formam o layout e a navegação do site.
  + **SessionProvider.tsx:** Envolve o aplicativo para gerenciar sessões com NextAuth.
* **/src/context:**
  + **CartContext.tsx:** Gerencia o estado do carrinho de compras (adição, remoção, atualização de quantidades), podendo persistir dados no localStorage.
* **next.config.js:**
  + Configurações do Next.js, como quais domínios são permitidos para exibir imagens (necessário para o componente <Image>).

**3. Fluxo de Dados e Funcionalidades**

**Backend**

* **Consulta de Produtos:**
  + O endpoint GET /api/products aceita diversos parâmetros de filtro (nome, categoria, subcategoria, marca, disponibilidade, preço mínimo e máximo) e parâmetros de paginação (page, limit).
  + A consulta é feita via Prisma no método getProducts, que monta o objeto where com as condições necessárias e utiliza skip e take para a paginação.
  + Retorna um objeto { products, total }.
* **Filtros:**
  + O endpoint GET /api/products/filters chama o método getFilterOptions do service, que utiliza o método groupBy do Prisma para obter valores únicos dos campos category, subcategory\_level\_1, brand e availability.
* **Serviço de Imagens:**
  + O ServeStaticModule está configurado no app.module.ts para servir arquivos estáticos do diretório imagestest2 sob a rota /images.
  + Assim, uma imagem armazenada em webmarket/backend/imagestest2/C00049/image\_1.jpg será acessível via URL:  
    http://<backend-dominio>/images/C00049/image\_1.jpg.
* **Início do Backend:**
  + O arquivo main.ts define o prefixo global "api", de forma que todos os endpoints do backend estejam sob /api (ex.: /api/products).

**Frontend**

* **Home Page (page.tsx):**
  + Exibe um banner com data atual e produtos destacados.
  + Apresenta uma lista de categorias que redirecionam para a página de produtos com o filtro aplicado.
  + Utiliza o componente ProductCard para exibir os produtos destacados.
* **Página de Produtos (products/page.tsx):**
  + Possui um painel lateral (sidebar) com filtros para busca, categoria, subcategoria, marca, disponibilidade e faixa de preço.
  + A área principal exibe os produtos usando o componente ProductCard e inclui paginação para navegar entre páginas.
  + A página faz chamadas à API do backend (usando fetch) e atualiza o estado conforme os filtros são aplicados.
* **Componentes Reutilizáveis:**
  + **ProductCard:** Exibe as informações dos produtos (imagem, nome, categoria, preço, disponibilidade) e inclui opções para adicionar ao carrinho.
  + Outros componentes (Navbar, Footer, Hero, etc.) formam a estrutura e navegação do site.
* **Gerenciamento do Carrinho:**
  + Através do CartContext, o estado do carrinho é gerenciado e persistido, permitindo que os usuários adicionem produtos e procedam para o checkout.

**4. Fluxo de Deploy (Para Demonstração)**

* **Backend:**
  + Configure as variáveis de ambiente (por exemplo, no arquivo .env.production).
  + Realize o build e inicie o servidor em modo de produção (npm run build e npm run start:prod).
  + Hospede em uma plataforma como Heroku, Render ou em um VPS com Docker.
* **Frontend:**
  + Configure as variáveis de ambiente (por exemplo, em .env.production), principalmente NEXT\_PUBLIC\_API\_URL.
  + Realize o build do Next.js (npm run build) e inicie o servidor (npm run start), ou utilize Vercel para deploy automático.
* **Banco de Dados:**
  + Hospede o PostgreSQL em um serviço gerenciado (Heroku Postgres, AWS RDS, etc.) e atualize a variável DATABASE\_URL.
* **Imagens:**
  + Se optar por servir as imagens via backend, garanta que o diretório imagestest2 esteja presente e que o ServeStaticModule esteja configurado corretamente.
  + Alternativamente, migre as imagens para um serviço de armazenamento na nuvem e atualize as URLs.

**5. Instruções de Teste e Validação**

* **Teste dos Endpoints do Backend:**
  + Acesse:
    - GET http://<backend-dominio>/api/products?page=1&limit=20
    - GET http://<backend-dominio>/api/products/filters
  + Verifique que os dados retornados estão corretos.
* **Teste do Frontend:**
  + Navegue até o domínio do frontend e verifique:
    - A home page exibe corretamente o banner, produtos destacados e categorias.
    - A página de produtos aplica os filtros e a paginação.
    - As imagens são carregadas corretamente.
    - A navegação entre páginas (produtos, detalhes, carrinho, checkout) está funcionando.
* **Depuração:**
  + Utilize o console do navegador e os logs do backend para identificar e corrigir quaisquer erros que possam surgir.
  + Teste os filtros individualmente e em combinação para garantir que os dados estão sendo filtrados corretamente.

**6. Considerações Finais**

Este resumo reúne todos os detalhes críticos do projeto "webmarket": a estrutura de arquivos, a funcionalidade dos endpoints, a integração entre backend e frontend e as instruções para deploy e testes. Você pode utilizar este documento para migrar o projeto para outro ambiente ou para documentar o estado atual do sistema.

Basta copiar o conteúdo do arquivo Markdown abaixo para um arquivo chamado, por exemplo, project\_summary.md, e você terá um registro completo e detalhado do estado e funcionamento do seu projeto.

# Projeto WebMarket – Resumo e Estado Atual

## Visão Geral

O WebMarket é um sistema de e-commerce construído com Next.js (frontend) e NestJS (backend) com acesso ao banco de dados PostgreSQL via Prisma. Ele oferece funcionalidades de autenticação, listagem de produtos com filtros (incluindo categoria, subcategoria, marca, disponibilidade, faixa de preço), paginação, e gerenciamento de carrinho/checkout.

## Estrutura de Arquivos

### Backend (webmarket/backend)

- \*\*/dist:\*\* Arquivos compilados.

- \*\*/imagestest2:\*\* Diretório com imagens dos produtos (ex.: C00049/image\_1.jpg).

- \*\*/node\_modules:\*\* Dependências do Node.js.

- \*\*/prisma/schema.prisma:\*\* Definição do modelo de dados (modelo "produtos").

- \*\*/products:\*\*

- \*products.controller.ts\* – Define os endpoints para consultar produtos e filtros.

- \*products.service.ts\* – Lógica de negócio para filtragem e paginação.

- \*products.module.ts\* – Conecta controller e service.

- \*\*/src:\*\*

- \*app.module.ts\* – Configura o aplicativo globalmente, importando o ProductsModule e o PrismaService; configura o ServeStaticModule para servir imagens.

- \*app.controller.ts, app.service.ts\* – Endpoints globais (se houver).

- \*main.ts\* – Inicializa o NestJS, define o prefixo global "api", habilita CORS e inicia o servidor na porta 4000.

- \*prisma.service.ts\* – Serviço para conexão com o banco via Prisma.

- \*\*.env:\*\* Variáveis de ambiente (DATABASE\_URL, etc.).

### Frontend (webmarket/frontend)

- \*\*.env.local:\*\* Variáveis de ambiente para desenvolvimento (NEXT\_PUBLIC\_API\_URL, NEXTAUTH\_URL, etc.).

- \*\*/public:\*\* Arquivos públicos (imagens, ícones).

- \*\*/src/app:\*\*

- \*page.tsx\* – Home page com banner, produtos destacados e categorias.

- \*products/page.tsx\* – Página de listagem de produtos com painel lateral de filtros, paginação e uso do componente ProductCard.

- \*products/[product\_code]/page.tsx\* – Página de detalhe do produto (com botão de adicionar ao carrinho e controle de quantidade).

- Outras páginas (auth, cart, checkout, dashboard, profile).

- \*\*/src/components:\*\*

- \*ProductCard.tsx\* – Componente reutilizável para exibir informações do produto.

- \*Navbar.tsx, Footer.tsx, Hero.tsx, Categories.tsx, FeaturedProducts.tsx, etc.\* – Componentes de layout e navegação.

- \*\*/src/context/CartContext.tsx:\*\* Gerenciamento do carrinho de compras com persistência (localStorage).

## Funcionalidades

- \*\*Autenticação:\*\* NextAuth para login/logout.

- \*\*Listagem de Produtos:\*\* Filtros dinâmicos (nome, categoria, subcategoria, marca, disponibilidade, preço) e paginação.

- \*\*Produto Detalhado:\*\* Exibe informações detalhadas e permite adicionar ao carrinho.

- \*\*Carrinho e Checkout:\*\* Gerenciamento e finalização de pedidos.

- \*\*Administração:\*\* Dashboard para controle de inventário e perfil do usuário.

## Endpoints do Backend

- `GET /api/products`: Retorna produtos com filtros e paginação (parâmetros: name, category, subcategory, brand, availability, minPrice, maxPrice, orderBy, order, page, limit).

- `GET /api/products/filters`: Retorna opções de filtro (categories, subcategories, brands, availabilities).

## Configuração de Imagens

- O ServeStaticModule no backend serve imagens a partir do diretório `imagestest2` sob a rota `/images`.

- Exemplo: Uma imagem em `imagestest2/C00049/image\_1.jpg` é acessível via `http://<backend-dominio>/images/C00049/image\_1.jpg`.

## Deploy

- \*\*Backend:\*\* Pode ser implantado em Heroku, Render, ou em um VPS/Docker. O backend usa um prefixo "api" e atende em `http://<backend-dominio>/api`.

- \*\*Frontend:\*\* Pode ser implantado no Vercel, Netlify, ou em um VPS. Configure as variáveis de ambiente para apontar para o backend.

## Testes e Depuração

- Utilize ferramentas como Postman para testar endpoints.

- Use o Prisma Studio (`npx prisma studio`) para verificar dados no banco.

- Monitore os logs do backend (NestJS e Prisma) e os do frontend (console do navegador).

---

Este documento contém os detalhes mais relevantes sobre o projeto WebMarket. Utilize-o para migrar ou documentar a situação atual do sistema.

Basta salvar o conteúdo acima como um arquivo de texto ou Markdown (por exemplo, project\_summary.md). Esse arquivo contém toda a informação necessária sobre o projeto, a estrutura, a funcionalidade e os passos para manutenção e eventual deploy.

Se precisar de mais alguma coisa, estou à disposição!