Pacote Completo — Loja MTA (Front-end + Back-end + MTA Script + DB + README)

> Abra este documento com atenção. Ele contém a estrutura do projeto, código pronto e instruções para deploy.

Estrutura do projeto

README (resumo)

Este pacote fornece um protótipo funcional para uma loja focada em produtos MTA com entrega automática no servidor. Inclui:

- Backend em Node.js/Express com endpoints para produtos, pedidos, autenticação.
- Integração com gateway de pagamentos (placeholders para Pix/MercadoPago/Stripe).
- Serviço que notifica o servidor MTA para entregar itens automaticamente via `fetchRemote` (no lado do MTA).
- Frontend em React (Vite) com loja, checkout e painel administrativo.
- Script Lua para MTA que recebe requisições seguras para entregar itens.
- Banco de dados (MySQL) com esquema base.

Banco de dados (MySQL) — migrations/schema.sql

Backend — .env.example

Backend — package.json (essenciais)

Backend — server.js (esqueleto)

Backend — src/models/index.js (Sequelize)

Backend — src/models/user.js

Backend — src/models/product.js

Backend — src/models/order.js

Backend — src/controllers/orderController.js (fluxo de pagamento -> notificação -> entrega)

Backend — src/services/mtaDeliveryService.js (chamada segura para MTA)

MTA — mta-script/mta_delivery.lua

> **Observação:** o método preferido é usar o `fetchRemote` do MTA em um resource que exponha um endpoint HTTP interno ou usar sockets seguros. Ajuste conforme sua infra.

Frontend — package.json (Vite + React)

Frontend — src/pages/Checkout.jsx (exemplo)

Segurança e recomendações importantes

1. **Nunca** exponha sua `MTA_API_KEY` no frontend. Use variáveis de ambiente no backend.
2. Use HTTPS em produção (certbot/Let's Encrypt).
3. Valide no backend que o jogador que receberá o item é realmente o dono do pedido — use username + id e checksums.
4. Proteja o webhook de pagamento usando assinatura HMAC do gateway.
5. Mantenha logs de `mta_logs` para auditar entregas e erros.
6. Em caso de falha de entrega automática, implemente rotina de retry e fallback para retirada manual.

README de deploy rápido

1. Criar banco: rodar `migrations/schema.sql` no MySQL.
2. Preencher `.env` no backend a partir de `.env.example`.
3. `cd backend && npm install` e `npm run dev`.
4. `cd frontend && npm install` e `npm run dev`.

5. Configure o servidor MTA para rodar `mta_delivery.lua` (ou adaptar para sua resource específica).
6. Configure webhook no gateway de pagamento para chamar `POST /api/orders/payment-webhook` do backend.

Observações finais

- Este pacote é um protótipo completo e deve ser adaptado ao seu servidor MTA (nomes de eventos, funções internas, forma de conceder VIP/coins etc.).
- Se quiser, eu posso:
- Gerar os arquivos reais em formato zip.
- Gerar código mais específico para o seu servidor (diga qual framework MTA usa, ou os eventos de VIP/coins do seu servidor).

Fim do documento.

Pacote Completo – Loja MTA com Retirada Presencial

A seguir está o pacote com todas as informações necessárias para criar um site de vendas focado em produtos de MTA, onde o cliente compra online e retira fisicamente na sua cidade.

---

1. Estrutura do Projeto

1. Front-end (site): - Página inicial - Página de produtos - Carrinho - Checkout - Painel do cliente (acompanhar pedidos) - Pagamento - Confirmação de retirada

2. Back-end: - Cadastro de produtos - Sistema de pedidos - Integração com banco de dados - Integração com pagamentos (ex: Pix, MercadoPago, PagBank) - API conectada ao servidor MTA (opcional)

3. Banco de dados: - Tabela usuários - Tabela produtos - Tabela pedidos - Tabela códigos / chaves MTA (opcional)

4. Integração com MTA: - Endpoint HTTP para comunicar com o servidor - Sistema interno no servidor para entregar item apenas após confirmação de retirada - Log de entrega

5. Sistema de Retirada Presencial: - Tela no painel ADM com lista de pedidos pendentes - Botão “Confirmar retirada” - Após retirada → liberar código / produto no MTA

---

2. Fluxo Completo do Usuário

1. Usuário acessa o site
2. Escolhe o produto (ex: VIP, moeda, item especial)

3. Faz o pagamento
4. Recebe o status “Aguardando retirada presencial”
5. Vai até o local físico
6. Você confirma a retirada pelo painel admin
7. O servidor MTA recebe a confirmação
8. O produto é entregue automaticamente no jogo

---

3. Arquivos Incluídos no Pacote

Front-end (HTML / CSS / JS):

- index.html
- produto.html
- carrinho.html
- checkout.html
- painel-usuario.html
- login.html
- admin.html
- estilos.css
- script.js

Back-end (Node.js):

- server.js
- routes/auth.js
- routes/produtos.js
- routes/pedidos.js
- routes/retirada.js
- config/db.js

Banco de Dados (MySQL):

- schema.sql (com todas as tabelas)

Integração MTA:

Resource: `shop_system` - server.lua - client.lua (se necessário) - config.lua

---

4. Requisitos Técnicos para subir o site

Servidor web:

- Hostinger / VPS / Localhost
- Node.js 18+
- PM2 (opcional)

Banco de dados:

- MySQL 5.7+ ou MariaDB

Integração MTA:

No seu servidor MTA você precisa habilitar: - `fetchRemote` - `callRemote` - ACL configurada para o resource aceitar requisições

---

---

7. Como você deve usar o pacote

1. Configurar domínio e hospedagem
2. Importar o banco de dados
3. Subir o back-end (Node.js)
4. Subir o front-end (arquivos estáticos)
5. Configurar o resource no MTA
6. Criar seu painel admin e cadastrar os produtos
7. Testar compra → retirada → entrega no jogo

Se quiser, eu também gero **os arquivos completos (ZIP)** ou **personalizo com seu nome, logo e produtos do seu servidor MTA.**

✔️Checklist Completo para Montar a Loja no Replit

1. Criar o projeto no Replit

- Criar um novo Repl → escolher **Node.js**
- Renomear para: `mta-shop`
- Criar as pastas:
- `/public`
- `/routes`
- `/config`
- `/database`

---

2. Estrutura obrigatória dentro do Replit

Pasta /public (páginas do site)

- index.html
- produtos.html
- carrinho.html

- checkout.html
- painel-usuario.html
- login.html
- admin.html
- estilos.css

Pasta /routes (funções do site)

- rotas de login
- rotas de produtos
- rotas de pedidos
- rotas de retirada

Pasta /config

- arquivo com configurações da loja
- chaves PIX (via variáveis de ambiente)
- opções de pagamento

Pasta /database

- arquivo de conexão com MySQL

✔️3. Configurar o banco de dados externo

Você precisa criar uma conta em algum serviço MySQL: - Hostinger - PlanetScale - Aiven - ClearDB

Depois: - Copiar a **URL de conexão** do banco - Colar no Replit em **Secrets → DATABASE_URL**

✔️4. Criar variáveis de ambiente no Replit

No menu **Secrets**, criar: - `DATABASE_URL` - `PIX_KEY` - `PAINEL_ADMIN_USER` - `PAINEL_ADMIN_PASS` - `MTA_API_KEY`

✔️5. Fluxo de funcionamento no Replit

1. Usuário entra no site
2. Escolhe produto
3. Compra
4. Pedido fica salvo como "Aguardando retirada"
5. Você confirma a retirada no painel admin
6. O Replit avisa o servidor MTA
7. Produto é entregue no jogo

✔️6. O que o MTA precisa para funcionar

• Um resource chamado **shop_system**
• Permitir chamadas HTTP ao seu Replit
• Um sistema interno para entregar o item
• Log de entrega

✔️7. Segurança obrigatória

• Criar API key secreta
• Não expor link de rotas internas
• Somente painel admin pode confirmar retirada
• Apenas o servidor MTA pode validar entregas

✔️8. Testes finais antes de liberar

• Testar compra
• Testar registro do pedido
• Testar painel admin
• Testar botão "Confirmar retirada"
• Ver se o MTA recebe a informação
• Ver se o item é entregue com segurança

Se quiser, posso gerar também um **passo a passo detalhado com instruções clicando, criando, nomeando cada parte do Replit**, totalmente leigo, se você pedir: *"quero o passo a passo detalhado"*.