#!/bin/bash

# Nombre del proyecto

PROJECT\_NAME="sebo"

echo "Creando estructura del proyecto '$PROJECT\_NAME'..."

# Crear directorio raíz del proyecto

mkdir -p "$PROJECT\_NAME"

cd "$PROJECT\_NAME"

# Crear directorios src, server, public, js, css

mkdir -p src/server/controllers src/server/utils src/public/js src/public/css

echo "Estructura de directorios creada."

# Crear archivos principales y agregar contenido

# package.json

cat << EOF > package.json

{

"name": "$PROJECT\_NAME",

"version": "1.0.0",

"description": "Project to monitor XRP exchange prices.",

"main": "src/server/app.js",

"scripts": {

"start": "node src/server/app.js",

"dev": "nodemon src/server/app.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"ccxt": "^4.3.56",

"cors": "^2.8.5",

"dotenv": "^16.4.5",

"express": "^4.19.2"

},

"devDependencies": {

"nodemon": "^3.1.2"

}

}

EOF

echo "package.json creado."

# .env.example

cat << EOF > .env.example

# Este archivo es un ejemplo. Renombra a .env y añade tus claves API

# para los exchanges que las requieran.

# Por ejemplo:

# KRAKEN\_API\_KEY=tu\_kraken\_api\_key

# KRAKEN\_SECRET=tu\_kraken\_secret

# BYNODEX\_API\_KEY=tu\_bynodex\_api\_key

# BYNODEX\_SECRET=tu\_bynodex\_secret

EOF

echo ".env.example creado."

# src/server/utils/config.js

cat << EOF > src/server/utils/config.js

require('dotenv').config();

const EXCHANGES = [

// Listado completo de los 42 exchanges de las imágenes

// Importante: Los nombres de los exchanges deben coincidir con los IDs de ccxt.

// Puedes encontrar los IDs de ccxt aquí: https://github.com/ccxt/ccxt/wiki/Exchange-IDs

// Para simplificar, he tomado los nombres directamente de las imágenes.

// Puede que necesites ajustar algunos nombres para que coincidan con ccxt.

// Por ejemplo, "Bynodex" podría ser "bynodex", "Coinbase Exchange" podría ser "coinbasepro" o "coinbase"

// He puesto los más comunes que identifiqué, necesitarías verificar para todos 42.

// Página 1 (imágenes 1, 4):

{ id: 'ascendex', name: 'AscendEX' },

{ id: 'binance', name: 'Binance' },

{ id: 'binanceth', name: 'Binance TH' }, // Posiblemente no directo en ccxt

{ id: 'binancetr', name: 'Binance TR' }, // Posiblemente no directo en ccxt

{ id: 'binanceus', name: 'Binance.US' },

{ id: 'bingx', name: 'BingX' },

{ id: 'bitfinex', name: 'Bitfinex' },

{ id: 'bitget', name: 'Bitget' },

{ id: 'bitrue', name: 'Bitrue' },

{ id: 'bitunix', name: 'Bitunix' },

// Página 2 (imágenes 2, 3):

{ id: 'btse', name: 'BTSE' },

{ id: 'bvox', name: 'BVOX' }, // Posiblemente no directo en ccxt

{ id: 'bybit', name: 'Bybit' },

{ id: 'bynoxdex', name: 'Bynodex' }, // Verificar ID exacto en ccxt

{ id: 'coinbasepro', name: 'Coinbase Exchange' }, // ccxt usa coinbasepro

{ id: 'coinex', name: 'CoinEx' },

{ id: 'coinw', name: 'CoinW' },

{ id: 'cryptocom', name: 'Crypto.com Exchange' }, // ccxt usa cryptocom

{ id: 'deepcoin', name: 'Deepcoin' },

{ id: 'digifinex', name: 'DigiFinex' },

// Página 3 (imagen con PancakeSwap, Phemex, Pionex, etc.)

{ id: 'pancakeswap', name: 'PancakeSwap v3 (BSC)' }, // ccxt tiene soporte limitado para DEXs

{ id: 'phemex', name: 'Phemex' },

{ id: 'pionex', name: 'Pionex' },

{ id: 'poloniex', name: 'Poloniex' },

{ id: 'probit', name: 'ProBit Global' }, // ccxt usa probit

{ id: 'tapbit', name: 'Tapbit' }, // Verificar ID exacto en ccxt

{ id: 'tokocrypto', name: 'Tokocrypto' },

{ id: 'toobit', name: 'Toobit' }, // Verificar ID exacto en ccxt

{ id: 'weex', name: 'WEEX' }, // Verificar ID exacto en ccxt

{ id: 'whitebit', name: 'WhiteBIT' },

// Página 4 (imagen con XT.COM, Zoomex)

{ id: 'xtcom', name: 'XT.COM' }, // ccxt usa xtcom

{ id: 'zoomex', name: 'Zoomex' }, // Verificar ID exacto en ccxt

// Faltan exchanges para llegar a 42, ya que las imágenes no muestran todos.

// Necesitarías completar esta lista con los nombres/IDs exactos de ccxt.

// Ejemplo de cómo se podrían añadir más:

// { id: 'kraken', name: 'Kraken' }, // Añadido manualmente si no estaba en tus imágenes

// { id: 'okx', name: 'OKX' },

// { id: 'kucoin', name: 'KuCoin' },

// ... hasta 42

];

module.exports = {

EXCHANGES,

PORT: process.env.PORT || 3000,

// Puedes agregar más configuraciones aquí, como rate limits globales

};

EOF

echo "src/server/utils/config.js creado."

# src/server/controllers/exchangeController.js

cat << 'EOF' > src/server/controllers/exchangeController.js

const ccxt = require('ccxt');

const { EXCHANGES } = require('../utils/config');

// Función para inicializar un exchange con ccxt

const initializeExchange = (exchangeId) => {

try {

// Asegúrate de que el ID del exchange es válido para ccxt

if (!ccxt.exchanges.includes(exchangeId)) {

console.warn(`[${exchangeId}] no es un ID de exchange válido para ccxt.`);

return null;

}

// Crear una instancia del exchange. Puedes añadir claves API aquí si las tienes.

// Por ejemplo:

// const api\_key = process.env[\`\${exchangeId.toUpperCase()}\_API\_KEY\`];

// const secret = process.env[\`\${exchangeId.toUpperCase()}\_SECRET\`];

//

// const exchangeConfig = {

// 'apiKey': api\_key,

// 'secret': secret,

// 'timeout': 10000, // Tiempo de espera para la respuesta

// 'enableRateLimit': true, // Habilitar la gestión de límites de tasa

// };

//

// return new ccxt[exchangeId](exchangeConfig);

// Para este ejemplo, solo inicializamos sin credenciales para probar conectividad pública

return new ccxt[exchangeId]({

'timeout': 10000,

'enableRateLimit': true,

});

} catch (error) {

console.error(`Error inicializando exchange ${exchangeId}: ${error.message}`);

return null;

}

};

// Función para obtener el estado y el precio de un exchange

const getExchangeStatusAndPrice = async (exchangeId, exchangeName) => {

const result = {

id: exchangeId,

name: exchangeName,

connected: false,

priceXRPUSDT: 'N/A',

error: null

};

const exchange = initializeExchange(exchangeId);

if (!exchange) {

result.error = `Failed to initialize ccxt for ${exchangeName}. Check if ID is correct or if ccxt supports it.`;

return result;

}

try {

// Intentar cargar los mercados para verificar conectividad básica

await exchange.loadMarkets();

result.connected = true;

// Intentar obtener el precio de XRP/USDT

if (exchange.markets['XRP/USDT']) {

const ticker = await exchange.fetchTicker('XRP/USDT');

result.priceXRPUSDT = ticker.last;

} else {

result.priceXRPUSDT = 'Pair not available';

}

} catch (e) {

result.connected = false;

result.error = e.message;

// console.error(`Error fetching data for ${exchangeName}: ${e.message}`);

}

return result;

};

// Endpoint para obtener el estado de todos los exchanges

const getExchangesStatus = async (req, res) => {

const statusPromises = EXCHANGES.map(ex => getExchangeStatusAndPrice(ex.id, ex.name));

const allExchangesStatus = await Promise.allSettled(statusPromises);

const formattedResults = allExchangesStatus.map(promiseResult => {

if (promiseResult.status === 'fulfilled') {

return promiseResult.value;

} else {

// Esto debería ser manejado por el catch dentro de getExchangeStatusAndPrice,

// pero es un fallback en caso de error Promise.allSettled

return {

id: 'unknown',

name: 'Unknown Exchange',

connected: false,

priceXRPUSDT: 'N/A',

error: promiseResult.reason ? promiseResult.reason.message : 'Unknown error'

};

}

});

res.json(formattedResults);

};

module.exports = {

getExchangesStatus,

};

EOF

echo "src/server/controllers/exchangeController.js creado."

# src/server/app.js

cat << EOF > src/server/app.js

const express = require('express');

const cors = require('cors');

const dotenv = require('dotenv');

const { PORT } = require('./utils/config');

const { getExchangesStatus } = require('./controllers/exchangeController');

dotenv.config();

const app = express();

// Middleware

app.use(cors()); // Habilita CORS para permitir peticiones desde el frontend

app.use(express.json()); // Permite a Express parsear JSON

// Servir archivos estáticos del frontend

app.use(express.static('src/public'));

// Endpoint para obtener el estado de los exchanges

app.get('/api/exchanges-status', getExchangesStatus);

// Ruta raíz para servir index.html (si alguien navega directamente al servidor)

app.get('/', (req, res) => {

res.sendFile(\_\_dirname + '/public/index.html');

});

// Iniciar el servidor

app.listen(PORT, () => {

console.log(\`Servidor Express corriendo en http://localhost:\${PORT}\`);

console.log('Accede al frontend en http://localhost:3000');

});

EOF

echo "src/server/app.js creado."

# src/public/index.html

cat << EOF > src/public/index.html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Monitoreo de Exchanges XRP</title>

<link rel="stylesheet" href="./css/style.css">

</head>

<body>

<div class="container">

<h1>Monitoreo de Exchanges XRP/USDT</h1>

<button id="refreshButton">Refrescar Estado</button>

<p>Los botones indican si se pudo establecer conexión y obtener datos de XRP/USDT.</p>

<div id="loading" class="loading">Cargando estado de exchanges...</div>

<div id="exchangeList" class="exchange-list">

</div>

</div>

<script src="./js/script.js"></script>

</body>

</html>

EOF

echo "src/public/index.html creado."

# src/public/js/script.js

cat << EOF > src/public/js/script.js

document.addEventListener('DOMContentLoaded', () => {

const exchangeListDiv = document.getElementById('exchangeList');

const refreshButton = document.getElementById('refreshButton');

const loadingDiv = document.getElementById('loading');

const fetchExchangeStatus = async () => {

exchangeListDiv.innerHTML = ''; // Limpiar lista anterior

loadingDiv.style.display = 'block'; // Mostrar spinner de carga

try {

const response = await fetch('http://localhost:3000/api/exchanges-status');

const data = await response.json();

loadingDiv.style.display = 'none'; // Ocultar spinner

if (data.length === 0) {

exchangeListDiv.innerHTML = '<p>No se encontraron exchanges o hubo un problema al cargar.</p>';

return;

}

data.forEach(ex => {

const exchangeItem = document.createElement('div');

exchangeItem.classList.add('exchange-item');

const statusDotClass = ex.connected ? 'status-dot-green' : 'status-dot-red';

const buttonText = ex.connected ? 'Conectado' : 'Fallo';

const priceText = ex.priceXRPUSDT !== 'N/A' && ex.priceXRPUSDT !== 'Pair not available'

? \`Precio XRP/USDT: \$ \${ex.priceXRPUSDT}\`

: ex.priceXRPUSDT === 'Pair not available'

? 'Par no disponible'

: 'Precio N/A';

exchangeItem.innerHTML = `

<h3>\${ex.name}</h3>

<div class="status-info">

<span class="status-dot \${statusDotClass}"></span>

<button class="status-button \${statusDotClass === 'status-dot-green' ? 'success' : 'failure'}">\${buttonText}</button>

</div>

<p class="price-info">\${priceText}</p>

\${ex.error ? \`<p class="error-info">Error: \${ex.error}</p>\` : ''}

`;

exchangeListDiv.appendChild(exchangeItem);

});

} catch (error) {

console.error('Error al obtener el estado de los exchanges:', error);

loadingDiv.style.display = 'none';

exchangeListDiv.innerHTML = '<p>Error al cargar la información. Inténtalo de nuevo más tarde.</p>';

}

};

refreshButton.addEventListener('click', fetchExchangeStatus);

// Cargar el estado al iniciar

fetchExchangeStatus();

});

EOF

echo "src/public/js/script.js creado."

# src/public/css/style.css

cat << EOF > src/public/css/style.css

body {

font-family: Arial, sans-serif;

background-color: #1a1a2e; /\* Dark background \*/

color: #e0e0e0; /\* Light text \*/

margin: 0;

padding: 20px;

display: flex;

justify-content: center;

min-height: 100vh;

box-sizing: border-box;

}

.container {

background-color: #2a2a4a; /\* Slightly lighter dark background \*/

padding: 30px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);

width: 100%;

max-width: 900px;

text-align: center;

}

h1 {

color: #8be9fd; /\* Light blue/cyan \*/

margin-bottom: 20px;

}

#refreshButton {

background-color: #6272a4; /\* Purple-gray \*/

color: white;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

margin-bottom: 20px;

transition: background-color 0.3s ease;

}

#refreshButton:hover {

background-color: #4a5c88;

}

.loading {

font-style: italic;

color: #f1fa8c; /\* Yellow \*/

margin-top: 20px;

}

.exchange-list {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));

gap: 20px;

margin-top: 30px;

}

.exchange-item {

background-color: #383a59; /\* Slightly darker purple \*/

padding: 15px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);

text-align: left;

display: flex;

flex-direction: column;

justify-content: space-between;

}

.exchange-item h3 {

margin-top: 0;

color: #ff79c6; /\* Pink \*/

font-size: 1.2em;

}

.status-info {

display: flex;

align-items: center;

gap: 10px;

margin-bottom: 10px;

}

.status-dot {

width: 12px;

height: 12px;

border-radius: 50%;

}

.status-dot-green {

background-color: #50fa7b; /\* Green \*/

}

.status-dot-red {

background-color: #ff5555; /\* Red \*/

}

.status-button {

border: none;

padding: 8px 12px;

border-radius: 5px;

color: white;

cursor: default; /\* Not clickable for status \*/

font-weight: bold;

text-transform: uppercase;

font-size: 0.8em;

}

.status-button.success {

background-color: #50fa7b; /\* Green \*/

}

.status-button.failure {

background-color: #ff5555; /\* Red \*/

}

.price-info {

font-size: 1.1em;

font-weight: bold;

color: #bd93f9; /\* Purple \*/

margin-top: 10px;

}

.error-info {

color: #ff6e6e; /\* Lighter red for errors \*/

font-size: 0.9em;

margin-top: 5px;

}

EOF

echo "src/public/css/style.css creado."

echo "Proyecto '$PROJECT\_NAME' y archivos iniciales creados exitosamente."

echo "Ahora, ejecuta 'cd $PROJECT\_NAME' y luego 'npm install' para instalar las dependencias."

echo "Después de instalar las dependencias, puedes ejecutar './run\_project.sh' para iniciar el servidor."