**Script del Conversor de Monedas**

**Descripción General**

El Conversor de Monedas es una aplicación web que permite a los usuarios convertir montos entre diferentes divisas, utilizando tasas de cambio en tiempo real. La aplicación está dividida en dos componentes principales:

1. **Backend**: Desarrollado con Node.js y Express. Se conecta a la API de Open Exchange Rates para obtener las tasas actualizadas.
2. **Frontend**: Desarrollado con Angular, proporciona una interfaz intuitiva y responsiva para realizar las conversiones.

**Arquitectura del Sistema**

**Tecnologías Utilizadas**

* **Backend**:
  + Node.js
  + Express
  + Axios (para peticiones HTTP)
  + dotenv (manejo de variables de entorno)
  + CORS (para habilitar peticiones desde el frontend)
* **Frontend**:
  + Angular 19
  + RxJS
  + Formularios Reactivos
  + CSS Responsivo

**Diagrama de Funcionamiento**

Usuario ↔ Frontend Angular ↔ Backend Express ↔ API Open Exchange Rates

**Funcionalidades Principales**

**1. Conversión de Monedas**

Permite convertir un monto desde una moneda origen a una moneda destino, mostrando:

* Resultado de la conversión
* Tasa de cambio actual entre ambas monedas

**2. Selección de Monedas**

El usuario puede elegir entre 15 monedas populares, incluyendo:

* USD, EUR, GBP, JPY, CAD, AUD, CHF, CNY, entre otras.

**3. Intercambio Rápido**

Un botón permite invertir fácilmente la moneda de origen con la de destino.

**4. Diseño Responsivo**

La interfaz se adapta a distintos dispositivos, ofreciendo una buena experiencia tanto en móviles como en escritorios.

**5. Validación de Formularios**

Se asegura que los datos ingresados sean válidos antes de proceder con la conversión.

**Detalles de Implementación**

**Backend (Node.js + Express)**

**Configuración del Servidor (server.js)**

const express = require('express');

const cors = require('cors');

const axios = require('axios');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

app.use(cors());

app.use(express.json());

const API\_KEY = process.env.OPENEXCHANGERATES\_API\_KEY;

**Endpoint de Conversión**

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

try {

const { from, to, amount } = req.query;

if (!from || !to || !amount) {

return res.status(400).json({ error: 'Parámetros from, to y amount son requeridos' });

}

const response = await axios.get(

`https://openexchangerates.org/api/latest.json?app\_id=${API\_KEY}&base=USD`

);

const rates = response.data.rates;

let convertedAmount;

if (from === 'USD') {

convertedAmount = amount \* rates[to];

} else {

const sourceRate = rates[from];

const targetRate = rates[to];

convertedAmount = (amount / sourceRate) \* targetRate;

}

res.json({

from,

to,

amount: parseFloat(amount),

result: convertedAmount,

rate: from === 'USD' ? rates[to] : rates[to] / rates[from]

});

} catch (error) {

console.error('Error en la conversión:', error);

res.status(500).json({ error: 'No se pudo obtener las tasas de cambio' });

}

});

**Frontend (Angular)**

**Servicio de Moneda (currency.service.ts)**

@Injectable({ providedIn: 'root' })

export class CurrencyService {

private apiUrl = 'http://localhost:5000/api';

constructor(private http: HttpClient) {}

convertCurrency(from: string, to: string, amount: number): Observable<ConversionResult> {

return this.http.get<ConversionResult>(

`${this.apiUrl}/convertir?from=${from}&to=${to}&amount=${amount}`

);

}

getCurrencies(): string[] {

return ['USD', 'EUR', 'GBP', 'JPY', 'CAD', 'AUD', 'CHF', 'CNY', 'HKD', 'MXN', 'INR', 'BRL', 'RUB', 'KRW', 'SGD'];

}

}

**Componente Principal (currency-converter.component.ts)**

@Component({

selector: 'app-currency-converter',

standalone: true,

imports: [CommonModule, ReactiveFormsModule],

templateUrl: './currency-converter.component.html',

styleUrls: ['./currency-converter.component.css']

})

export class CurrencyConverterComponent implements OnInit {

converterForm: FormGroup;

currencies: string[] = [];

result: ConversionResult | null = null;

loading = false;

error = '';

constructor(private fb: FormBuilder, private currencyService: CurrencyService) {

this.converterForm = this.fb.group({

amount: [1, [Validators.required, Validators.min(0.01)]],

fromCurrency: ['USD', Validators.required],

toCurrency: ['EUR', Validators.required]

});

}

ngOnInit(): void {

this.currencies = this.currencyService.getCurrencies();

}

convertCurrency(): void {

if (this.converterForm.invalid) return;

const { amount, fromCurrency, toCurrency } = this.converterForm.value;

this.error = '';

this.loading = true;

this.currencyService.convertCurrency(fromCurrency, toCurrency, amount).subscribe({

next: (data) => {

this.result = data;

this.loading = false;

},

error: (err) => {

this.error = 'Error al convertir la moneda. Intente nuevamente.';

this.loading = false;

console.error('Error:', err);

}

});

}

}

**Plantilla HTML**

Incluye:

* Formulario con campos de cantidad, moneda origen y destino
* Botón para intercambiar monedas
* Botón de conversión
* Secciones para mostrar resultado o errores

**Estilos CSS Responsivos**

@media (max-width: 576px) {

.converter-container {

padding: 1.5rem;

}

.currency-selectors {

flex-direction: column;

}

.swap-btn {

margin: 0.5rem 0;

transform: rotate(90deg);

}

}

**Manejo de Errores**

* Validaciones en el formulario (evita valores negativos o vacíos)
* Manejo de errores HTTP en frontend y backend
* Mensajes de error amigables para el usuario

**Consideraciones Finales**

**Limitaciones**

* La API gratuita de Open Exchange Rates limita a 1,000 peticiones mensuales y solo permite USD como base.

**Seguridad**

* La clave API está protegida mediante variables de entorno.
* Se recomienda implementar autenticación en entornos de producción.

**Escalabilidad**

* Se puede mejorar el rendimiento implementando almacenamiento en caché o bases de datos para tasas recientes.

**Conclusión**

Esta aplicación demuestra la integración eficiente entre un backend Node.js y un frontend Angular, haciendo uso de una API externa para obtener información en tiempo real. Su estructura modular facilita el mantenimiento y su diseño responsivo mejora la experiencia del usuario. Es una base sólida para futuras extensiones y mejoras.