

Informe sobre el libro

las riquezas de las naciones adam smith

App cap 4, 5, 6

jhoan bernal

30 de junio de 2025

Capítulo 4: Del origen y uso del dinero

Conceptos Clave:

Problemas del trueque:

"Doble coincidencia de necesidades": Para intercambiar, ambas partes deben querer lo que el otro ofrece.

Dificultad para dividir ciertos bienes (ej: una vaca por herramientas).

Surgimiento del dinero:

Metales preciosos (oro, plata) se convirtieron en medio de intercambio universal.

Acuñaación de monedas para garantizar peso y pureza.

Funciones del dinero:

Medio de intercambio

Depósito de valor

Unidad de cuenta

```
// Moneda virtual basada en metales preciosos
class Moneda {
    constructor(nombre, valorRelativoOro) {
        this.nombre = nombre;
        this.valor = valorRelativoOro;
        this.historialValores = [];
    }
}
```

```
actualizarValor(nuevoValor) {  
    this.historialValores.push(this.valor);  
    this.valor = nuevoValor;  
}  
}  
  
const monedas = {  
    "oro": new Moneda("Onza de Oro", 1),  
    "plata": new Moneda("Onza de Plata", 0.05),  
    "cobre": new Moneda("Libra de Cobre", 0.001)  
};
```

Implementa en tu app:

Usuarios pueden elegir en qué moneda mantener sus ahorros

Sistema de conversión automática entre metales

Historial de valores de las monedas

Paso 1: Ampliación del Modelo de Usuario

¿Qué hicimos?

- Modificamos la estructura del inventario de cada usuario para que ahora, además de trigo, herramientas y dinero, también tenga:
- Oro (oro): 1 onza de oro = 1 hora de trabajo

- Plata (plata): 1 onza de plata = 0.05 horas de trabajo
- Cobre (cobre): 1 libra de cobre = 0.001 horas de trabajo

Esto permite que cada usuario tenga saldos separados en oro, plata y cobre, además del dinero tradicional.

¿Por qué es importante?

- Así podemos simular cómo los usuarios pueden elegir en qué moneda guardar sus ahorros y cómo el valor de cada moneda puede fluctuar, tal como lo explica Adam Smith en el Capítulo 4.

db.go

```
Oro          float64 `json:"oro"`    // 1 onza de oro = 1
hora de trabajo

Plata        float64 `json:"plata"`  // 1 onza de plata =
0.05 horas de trabajo

Cobre        float64 `json:"cobre"`  // 1 libra de cobre
= 0.001 horas de trabajo

// Estructura para monedas y su historial
type Moneda struct {

    Nombre          string    `json:"nombre"`

    ValorRelativoOro float64    `json:"valor_relativo_oro"`

    HistorialValores []float64 `json:"historial_valores"`

}
```

```

// Mapa global de monedas

var Monedas = map[string]*Moneda{

    "oro":    {Nombre: "Onza de Oro", ValorRelativoOro: 1,
HistorialValores: []float64{1}},

    "plata": {Nombre: "Onza de Plata", ValorRelativoOro: 0.05,
HistorialValores: []float64{0.05}},

    "cobre": {Nombre: "Libra de Cobre", ValorRelativoOro: 0.001,
HistorialValores: []float64{0.001}},

}

// Función para actualizar el valor de una moneda y guardar
historial

func ActualizarValorMoneda(nombre string, nuevoValor float64) {

    if moneda, ok := Monedas[nombre]; ok {

        moneda.HistorialValores =
append(moneda.HistorialValores, moneda.ValorRelativoOro)

        moneda.ValorRelativoOro = nuevoValor

    }

}

// Función de conversión entre monedas

func ConvertirMoneda(cantidad float64, monedaOrigen,
monedaDestino string) float64 {

```

```
m1, ok1 := Monedas[monedaOrigen]

m2, ok2 := Monedas[monedaDestino]

if !ok1 || !ok2 || m2.ValorRelativoOro == 0 {

    return 0

}

return cantidad * m1.ValorRelativoOro / m2.ValorRelativoOro
}
```

Paso 2: Estructura y Lógica de Monedas

¿Qué agregamos?

- Una estructura llamada Moneda que representa cada tipo de moneda (oro, plata, cobre) y guarda:
- Su nombre
- Su valor relativo al oro
- Un historial de valores (para ver cómo cambia con el tiempo)
- Un mapa global Monedas para acceder fácilmente a cada moneda.
- Funciones para:
- Actualizar el valor de una moneda y guardar el historial.
- Convertir entre monedas usando sus valores relativos.

¿Por qué es importante?

- Así podemos simular la conversión entre monedas y mostrar cómo el valor de cada una puede cambiar con el tiempo, permitiendo a los usuarios ver el historial y tomar decisiones.

¿Qué sigue?

Siguiente paso: Vamos a crear los primeros endpoints REST para que puedas:

1. Consultar los valores actuales de las monedas.
1. Consultar el historial de valores de cada moneda.
1. Realizar conversiones entre monedas.

```
// ===== ENDPOINTS DEL SISTEMA DE MONEDAS (CAPÍTULO 4) =====

// Endpoint para consultar los valores actuales de las
monedas

router.GET("/api/monedas/valores", func(c *gin.Context) {

    valores := make(map[string]gin.H)

    for k, v := range database.Monedas {

        valores[k] = gin.H{

            "nombre":          v.Nombre,

            "valor_relativo_oro": v.ValorRelativoOro,

        }

    }

    c.JSON(http.StatusOK, gin.H{

        "success": true,

        "valores": valores,

    })

})
```

```
// Endpoint para consultar el historial de valores de una
moneda

router.GET("/api/monedas/historial/:moneda", func(c
*gin.Context) {

    moneda := c.Param("moneda")

    m, ok := database.Monedas[moneda]

    if !ok {

        c.JSON(http.StatusNotFound, gin.H{

            "success": false,

            "error": "Moneda no encontrada",

        })

        return

    }

    c.JSON(http.StatusOK, gin.H{

        "success": true,

        "historial": m.HistorialValores,

    })

})

// Endpoint para convertir entre monedas
```



```
router.GET("/api/monedas/convertir/:cantidad/:origen/:destino",
func(c *gin.Context) {

    cantidadStr := c.Param("cantidad")

    origen := c.Param("origen")

    destino := c.Param("destino")

    cantidad, err := strconv.ParseFloat(cantidadStr, 64)

    if err != nil {

        c.JSON(http.StatusBadRequest, gin.H{

            "success": false,

            "error": "Cantidad inválida",

        })

        return

    }

    resultado := database.ConvertirMoneda(cantidad, origen,
destino)

    c.JSON(http.StatusOK, gin.H{

        "success": true,

        "cantidad_origen": cantidad,

        "moneda_origen": origen,

        "moneda_destino": destino,

        "resultado": resultado,
```

```
    })
  })
}
```

¿Qué endpoints se agregaron?

1. Consultar valores actuales de las monedas
 - GET /api/monedas/valores
 - Devuelve el valor actual de oro, plata y cobre respecto al oro.
1. Consultar el historial de valores de una moneda
 - GET /api/monedas/historial/:moneda
 - Ejemplo: /api/monedas/historial/oro
 - Devuelve el historial de valores de la moneda seleccionada.
1. Realizar conversiones entre monedas
 - GET /api/monedas/convertir/:cantidad/:origen/:destino
 - Ejemplo: /api/monedas/convertir/10/oro/plata
 - Devuelve cuánta plata obtienes por 10 onzas de oro (o cualquier otra combinación).

paso 3: ¿Qué contendrá la página /dinero?

1. Panel de conversión de monedas
 - Selecciona cantidad, moneda origen y destino, y muestra el resultado.
1. Visualización de valores actuales y gráficos de historial
 - Tabla con valores actuales de oro, plata y cobre.
 - Gráfica del historial de cada moneda.
1. Panel de usuarios y sus saldos en cada moneda
 - Lista de usuarios y cuánto tienen en oro, plata, cobre y dinero.

1. Explicación visual de los problemas del trueque vs. el uso del dinero

- Breve resumen y visualización educativa.

```
<!DOCTYPE html>

<html lang="es">

  <head>

    <meta charset="UTF-8" />

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

    <title>Simulador del Dinero - Capítulo 4</title>

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

    <link rel="stylesheet" href="/assets/css/dinero.css" />

    <link
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/
css/all.min.css" rel="stylesheet" />

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

  </head>

  <body>

    <nav>

      <div class="logo">

        <h1>LOGO</h1>

      </div>
```

```
<ul>

  <li>

    <a href="/expansion">Cap Anterior</a>

  </li>

  <li>

    <a href="/dinero">Cap 4: El origen y uso del
dinero</a>

  </li>

  <li>

    <a href="/">Cap 5:</a>

  </li>

  <li>

    <a href="/">Cap 6:</a>

  </li>

  <li>

    <a href="/dinero">Siguietes Cap</a>

  </li>

</ul>

<div class="hamburger">

  <span class="line"></span>

  <span class="line"></span>
```

```
<span class="line"></span>

</div>

</nav>

<div class="menubar">

  <ul>

    <li>

      <a href="#">Cap 1 fábrica de alfileres</a>

    </li>

    <li>

      <a href="/trueque">Cap 2 Sistema de trueque
inteligente</a>

    </li>

    <li>

      <a href="/expansion">Cap 3 Módulo de expansión de
mercado</a>

    </li>

    <li>

      <a href="/dinero">Siguietes Cap</a>

    </li>

  </ul>

</div>
```

```

<div class="container dinero-container">

  <!-- Bloque 1: Teoría -->

  <section class="dinero-bloque teoria">

    <h2><i class="fas fa-coins"></i> Capítulo 4: Del origen
y uso del dinero</h2>

    <div class="teoria-content">

      <div class="teoria-col">

        <h3>Problemas del trueque</h3>

        <ul>

          <li>

            <b>Doble coincidencia de necesidades:</b> Para
intercambiar, ambas partes deben querer lo que el otro ofrece.

          </li>

          <li>

            <b>Dificultad para dividir bienes:</b> Ejemplo:
una vaca por herramientas.

          </li>

        </ul>

      </div>

      <div class="teoria-col">

        <h3>Surgimiento del dinero</h3>

        <ul>

```

```
        <li>Metales preciosos (oro, plata) como medio de  
intercambio universal.</li>
```

```
        <li>Acuñación de monedas para garantizar peso y  
pureza.</li>
```

```
    </ul>
```

```
</div>
```

```
<div class="teoria-col">
```

```
    <h3>Funciones del dinero</h3>
```

```
    <ul>
```

```
        <li>Medio de intercambio</li>
```

```
        <li>Depósito de valor</li>
```

```
        <li>Unidad de cuenta</li>
```

```
    </ul>
```

```
</div>
```

```
</div>
```

```
</section>
```

```
<!-- Bloque 2: Conversión de monedas -->
```

```
<section class="dinero-bloque conversion">
```

```
    <h2><i class="fas fa-exchange-alt"></i> Conversión de  
monedas</h2>
```

```
    <form id="conversionForm" class="conversion-form">
```

```
<input type="number" id="cantidadInput" min="0"
step="any" placeholder="Cantidad" required />

<select id="origenSelect">

    <option value="oro">Oro</option>

    <option value="plata">Plata</option>

    <option value="cobre">Cobre</option>

</select>

<span class="arrow"><i class="fas
fa-arrow-right"></i></span>

<select id="destinoSelect">

    <option value="oro">Oro</option>

    <option value="plata">Plata</option>

    <option value="cobre">Cobre</option>

</select>

<button type="submit" class="btn btn-primary"><i
class="fas fa-sync-alt"></i> Convertir</button>

</form>

<div id="conversionResultado"
class="conversion-resultado"></div>

</section>

<!-- Bloque 3: Valores actuales de monedas -->
```



```

    <section class="dinero-bloque valores">

        <h2><i class="fas fa-balance-scale"></i> Valores
actuales de las monedas</h2>

        <table class="tabla-monedas">

            <thead>

                <tr>

                    <th>Moneda</th>

                    <th>Valor relativo al oro</th>

                </tr>

            </thead>

            <tbody id="tablaValoresMonedas">

                <!-- Se llena dinámicamente -->

            </tbody>

        </table>

    </section>

    <!-- Bloque 4: Historial de valores de monedas -->

    <section class="dinero-bloque historial">

        <h2><i class="fas fa-chart-line"></i> Historial de
valores de las monedas</h2>

        <div class="historial-graficas">

```

```

    <div class="grafica-moneda">

        <h4>Oro</h4>

        <canvas id="graficaOro"></canvas>

    </div>

    <div class="grafica-moneda">

        <h4>Plata</h4>

        <canvas id="graficaPlata"></canvas>

    </div>

    <div class="grafica-moneda">

        <h4>Cobre</h4>

        <canvas id="graficaCobre"></canvas>

    </div>

</div>

</section>

<!-- Bloque 5: Usuarios y sus saldos -->

<section class="dinero-bloque usuarios">

    <h2><i class="fas fa-users"></i> Usuarios y sus saldos
en monedas</h2>

    <table class="tabla-usuarios">

        <thead>

```

```

        <tr>

            <th>Nombre</th>

            <th>Oro</th>

            <th>Plata</th>

            <th>Cobre</th>

            <th>Dinero</th>

        </tr>

    </thead>

    <tbody id="tablaUsuariosMonedas">

        <!-- Se llena dinámicamente -->

    </tbody>

</table>

</section>

</div>

<script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>

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

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

</body>

</html>

```

La estructura HTML para la página /dinero está creada, con bloques verticales segmentados para:

- 1. Teoría (problemas del trueque, surgimiento y funciones del dinero)
- 1. Conversión de monedas
- 1. Tabla de valores actuales de monedas
- 1. Gráficas de historial de monedas
- 1. Tabla de usuarios y sus saldos en cada moneda

creamos el dinero.css

```
/* ===== Estilos para Simulador del Dinero (Capítulo 4) ===== */

.dinero-container {

    max-width: 1200px;

    margin: 40px auto;

    padding: 0 20px;

    display: flex;

    flex-direction: column;

    gap: 40px;

}

.dinero-bloque {

    background: #fff;

    border-radius: 16px;

    box-shadow: 0 4px 24px rgba(102,126,234,0.08);
```

```
padding: 32px 28px;

margin-bottom: 0;

display: flex;

flex-direction: column;

gap: 24px;
}

.dinero-bloque h2 {

    color: #764ba2;

    font-size: 1.5rem;

    margin-bottom: 12px;

    display: flex;

    align-items: center;

    gap: 10px;
}

/* Teoría */

.teoria-content {

    display: flex;

    gap: 32px;

    flex-wrap: wrap;
```

```
        justify-content: space-between;
    }

    .teoria-col {

        flex: 1 1 220px;

        background: linear-gradient(135deg, #f093fb 0%, #f5576c
100%);

        color: #fff;

        border-radius: 12px;

        padding: 18px 20px;

        min-width: 200px;

        box-shadow: 0 2px 12px rgba(240,147,251,0.08);

    }

    .teoria-col h3 {

        margin-bottom: 10px;

        font-size: 1.1rem;

    }

    .teoria-col ul {

        margin: 0;

        padding-left: 18px;

        font-size: 0.98rem;

    }
}
```

```
/* Conversión de monedas */

.conversion-form {

    display: flex;

    align-items: center;

    gap: 12px;

    flex-wrap: wrap;

}

.conversion-form input, .conversion-form select {

    padding: 10px 14px;

    border-radius: 8px;

    border: 1px solid #e9ecef;

    font-size: 1rem;

    min-width: 90px;

}

.conversion-form .arrow {

    font-size: 1.3rem;

    color: #764ba2;

}

.conversion-form .btn {

    padding: 10px 20px;
```

```
background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);

color: #fff;

border: none;

border-radius: 8px;

font-weight: 600;

cursor: pointer;

transition: box-shadow 0.2s;
}

.conversion-form .btn:hover {

    box-shadow: 0 2px 12px rgba(102,126,234,0.18);
}

.conversion-resultado {

    margin-top: 16px;

    font-size: 1.1rem;

    color: #333;

    font-weight: 500;
}

/* Tabla de valores de monedas */

.tabla-monedas {
```



```
width: 100%;

border-collapse: collapse;

margin-top: 10px;
}

.tabla-monedas th, .tabla-monedas td {

padding: 12px 16px;

border-bottom: 1px solid #e9ecef;

text-align: center;
}

.tabla-monedas th {

background: #f8f9fa;

color: #764ba2;

font-weight: 700;
}

/* Gráficas de historial */

.historial-graficas {

display: flex;

gap: 32px;

flex-wrap: wrap;

justify-content: space-between;
```

```
}

.grafica-moneda {

    flex: 1 1 250px;

    background: #f8f9fa;

    border-radius: 12px;

    padding: 18px 12px 12px 12px;

    min-width: 220px;

    box-shadow: 0 2px 12px rgba(102,126,234,0.06);

    display: flex;

    flex-direction: column;

    align-items: center;

}

.grafica-moneda h4 {

    color: #764ba2;

    margin-bottom: 10px;

    font-size: 1.05rem;

}

.grafica-moneda canvas {

    width: 100% !important;

    max-width: 220px;

    height: 160px !important;
```

```
}

/* Tabla de usuarios */

.tabla-usuarios {

    width: 100%;

    border-collapse: collapse;

    margin-top: 10px;

}

.tabla-usuarios th, .tabla-usuarios td {

    padding: 12px 16px;

    border-bottom: 1px solid #e9ecef;

    text-align: center;

}

.tabla-usuarios th {

    background: #f8f9fa;

    color: #6677aa;

    font-weight: 700;

}

/* Responsive */

@media (max-width: 900px) {
```

```
.teoria-content, .historial-graficas {  
    flex-direction: column;  
    gap: 18px;  
}  
  
.grafica-moneda {  
    min-width: 160px;  
}  
}  
  
@media (max-width: 600px) {  
    .dinero-bloque {  
        padding: 18px 6px;  
    }  
  
    .teoria-col {  
        padding: 12px 8px;  
    }  
  
    .grafica-moneda {  
        padding: 10px 2px 2px 2px;  
    }  
  
    .tabla-monedas th, .tabla-monedas td, .tabla-usuarios th,  
.tabla-usuarios td {
```

```
padding: 7px 4px;

font-size: 0.95rem;

}

}
```

El archivo dinero.css ya está creado con un diseño moderno, claro y responsivo para la página /dinero. Siguiente paso: Voy a crear el archivo JavaScript (dinero.js) para que la página funcione dinámicamente:

- Conversión de monedas
- Carga de valores actuales
- Gráficas de historial
- Tabla de usuarios y sus saldos

creamos el [dinero.js](#)

```
// JS para Simulador del Dinero (Capítulo 4)

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

    cargarValoresMonedas();

    cargarUsuarios();

    inicializarGraficas();

    setupConversion();

});
```

```
});

// Cargar valores actuales de monedas y llenar tabla
async function cargarValoresMonedas() {

    try {

        const res = await fetch('/api/monedas/valores');

        const data = await res.json();

        if (!data.success) throw new Error('No se pudo obtener valores de monedas');

        const tbody =
document.getElementById('tablaValoresMonedas');

        tbody.innerHTML = '';

        for (const key of ['oro', 'plata', 'cobre']) {

            if (data.valores[key]) {

                const row = document.createElement('tr');

                row.innerHTML = `

                    <td>${data.valores[key].nombre}</td>

<td>${data.valores[key].valor_relativo_oro}</td>

                `;

                tbody.appendChild(row);

            }

        }

    }

}
```

```

    }

    } catch (err) {

        mostrarError('Error al cargar valores de monedas');

    }

}

// Cargar usuarios y mostrar sus saldos en monedas
async function cargarUsuarios() {

    try {

        const res = await fetch('/api/users');

        const data = await res.json();

        if (!data.success) throw new Error('No se pudo obtener usuarios');

        const tbody =
document.getElementById('tablaUsuariosMonedas');

        tbody.innerHTML = '';

        data.users.forEach(u => {

            const inv = u.inventario;

            const row = document.createElement('tr');

            row.innerHTML = `

                <td>${u.nombre}</td>

```

```

        <td>${inv.oro ? inv.oro : 0}</td>

        <td>${inv.plata ? inv.plata : 0}</td>

        <td>${inv.cobre ? inv.cobre : 0}</td>

        <td>${inv.dinero ? inv.dinero : 0}</td>

        `;

        tbody.appendChild(row);

    });

} catch (err) {

    mostrarError('Error al cargar usuarios');

}

}

// Configurar formulario de conversión

function setupConversion() {

    const form = document.getElementById('conversionForm');

    form.addEventListener('submit', async (e) => {

        e.preventDefault();

        const cantidad =
parseFloat(document.getElementById('cantidadInput').value);

        const origen =
document.getElementById('origenSelect').value;

```



```
const destino =
document.getElementById('destinoSelect').value;

const resultadoDiv =
document.getElementById('conversionResultado');

resultadoDiv.textContent = '';

if (isNaN(cantidad) || cantidad < 0) {

    mostrarError('Cantidad inválida');

    return;

}

if (origen === destino) {

    resultadoDiv.textContent = 'Selecciona monedas
diferentes para convertir.';

    return;

}

try {

    const res = await
fetch(`/api/monedas/convertir/${cantidad}/${origen}/${destino}`)
;

    const data = await res.json();

    if (!data.success) throw new Error('No se pudo
convertir');

    resultadoDiv.textContent = `${cantidad} ${origen}
equivalen a ${data.resultado.toFixed(4)} ${destino}`;
```

```

        } catch (err) {

            mostrarError('Error en la conversión');

        }

    });
}

// Inicializar y cargar gráficas de historial
function inicializarGraficas() {

    cargarHistorialYGraficar('oro', 'graficaOro', '#FFD700');

    cargarHistorialYGraficar('plata', 'graficaPlata',
'#C0C0C0');

    cargarHistorialYGraficar('cobre', 'graficaCobre',
'#B87333');

}

async function cargarHistorialYGraficar(moneda, canvasId, color)
{

    try {

        const res = await
fetch(`/api/monedas/historial/${moneda}`);

        const data = await res.json();

        if (!data.success) throw new Error('No se pudo obtener
historial');
    }
}

```

```
const ctx =
document.getElementById(canvasId).getContext('2d');

// Si ya existe una gráfica previa, destrúyela
if (ctx.chartInstance) {

    ctx.chartInstance.destroy();

}

ctx.chartInstance = new Chart(ctx, {

    type: 'line',

    data: {

        labels: data.historial.map((_, i) => `T${i+1}`),

        datasets: [{

            label: `Valor de ${moneda}`,

            data: data.historial,

            borderColor: color,

            backgroundColor: color + '22',

            fill: true,

            tension: 0.3

        }]

    },

    options: {

        responsive: true,
```

```

        plugins: { legend: { display: false } },

        scales: {

            y: { beginAtZero: true }

        }

    });

} catch (err) {

    mostrarError(`Error al cargar historial de ${moneda}`);

}

}

// Mostrar errores de forma clara

function mostrarError(msg) {

    alert(msg);

}

```

el archivo dinero.js ya está creado y preparado para:

- Cargar y mostrar los valores actuales de las monedas.
- Permitir la conversión entre monedas y mostrar el resultado.
- Cargar y graficar el historial de valores de cada moneda (usando Chart.js).
- Cargar y mostrar los saldos de los usuarios en oro, plata, cobre y dinero.
- Manejar errores y validaciones para evitar fallos.

Capítulo 5: Del precio real y nominal de las mercancías

Conceptos Clave:

Teoría del valor-trabajo:

Valor real = Cantidad de trabajo requerido para producir un bien

Valor nominal = Cantidad de dinero que representa

Distorsiones monetarias:

La inflación afecta el valor nominal pero no el real

Ejemplo histórico: Plata americana redujo el valor nominal en Europa

Ejemplo aplicación en tu App:

Sistema de precios dual:

```
class Producto:

    def __init__(self, nombre, horas_trabajo):

        self.nombre = nombre

        self.horas_trabajo = horas_trabajo # Valor real

        self.precio_moneda = 0 # Valor nominal

    def actualizar_precio(self, valor_moneda):

        # Convierte valor real a nominal

        self.precio_moneda = self.horas_trabajo * valor_moneda
```

Ejemplo:

```
trigo = Producto("Trigo", 2) # 2 horas de trabajo por unidad
trigo.actualizar_precio(monedas["oro"].valor) # 2 * 1 = 2 onzas
de oro
```

Dashboard de inflación:

Muestra cómo cambia el poder adquisitivo de las monedas

Compara precios reales vs. nominales históricos

Usando api de [API | Datos 360 del Banco Mundial \(worldbank.org\)](https://data.worldbank.org/)

Plan de acción para pruebas con la API del World Bank Data360:


1 ¿Cómo haremos la prueba?

- Crearemos un archivo nuevo en una carpeta de pruebas, por ejemplo:

cap1_division_del_trabajo/cap1_division_del_trabajo/internal/experiments/worldbank_api_demo.
go

- El archivo será un pequeño programa o test que:
- Consulta la API pública del World Bank (por ejemplo, para inflación/CPI o precios de commodities).
- Imprime los resultados en consola para que puedas ver cómo se reciben y procesan los datos.
- No modificaremos la lógica principal ni los endpoints de tu app hasta que la integración esté probada.

2 ¿Qué probaremos primero?



empezar con el Índice de Precios al Consumidor (CPI), que es el dato más usado para inflación y está disponible en la API del World Bank.

3 ¿Cómo será el flujo?

- Usaremos el paquete estándar net/http de Go para hacer la petición.
- Procesaremos la respuesta JSON.
- Mostraremos los datos relevantes (por ejemplo, inflación anual de Colombia o el país que elijas).

4 ¿Dónde lo guardamos?

- Carpeta: internal/experiments/
- Archivo: worldbank_api_demo.go

codigo

```

package main

import (
    "encoding/json"
    "fmt"
    "io/ioutil"
    "net/http"
    "strconv"

)

// Estructura para parsear la respuesta del World Bank
// La respuesta es un array: [metadata, [data...]]
type WBResponse []interface{}

// Estructura para parsear cada registro de datos
type CPIRecord struct {
    Country struct {
        ID    string `json:"id"`
        Value string `json:"value"`
    } `json:"country"`
    Date string `json:"date"`
}

```



```
Value interface{} `json:"value"` // Usamos interface{} para
manejar diferentes tipos
}

// Función para extraer el valor numérico de manera segura
func (c *CPIRecord) GetNumericValue() (float64, bool) {

    if c.Value == nil {

        return 0, false

    }

    switch v := c.Value.(type) {

    case float64:

        return v, true

    case int:

        return float64(v), true

    case string:

        if v == "" {

            return 0, false

        }

        if num, err := strconv.ParseFloat(v, 64); err == nil {

            return num, true

        }

    }

}
```

```

    }

    return 0, false

default:

    return 0, false

}

}

func main() {

    // Venezuela: country code 'VE', Colombia: 'CO'

    countryCodes := []string{"VE", "CO"}

    indicator := "FP.CPI.TOTL" // CPI (Consumer Price Index)

    for _, country := range countryCodes {

        fmt.Printf("\n=== CPI para %s ===\n", country)

        url :=
fmt.Sprintf("https://api.worldbank.org/v2/country/%s/indicator/%s?format=json&per_page=10", country, indicator)

        resp, err := http.Get(url)

        if err != nil {

            fmt.Printf("Error en la petición para %s: %v\n",
country, err)

```

```
        continue

    }

    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)

    if err != nil {

        fmt.Printf("Error leyendo respuesta para %s: %v\n",
country, err)

        continue

    }

    var wbResp WBResponse

    if err := json.Unmarshal(body, &wbResp); err != nil {

        fmt.Printf("Error parseando JSON para %s: %v\n",
country, err)

        continue

    }

    if len(wbResp) < 2 {

        fmt.Printf("Respuesta inesperada para %s: %s\n",
country, string(body))

        continue

    }

}
```

```

    }

    // El segundo elemento es el array de datos
    recordsRaw, ok := wbResp[1].([]interface{})

    if !ok {

        fmt.Printf("Formato de datos inesperado para %s\n",
country)

        continue
    }

    fmt.Printf("Registros encontrados: %d\n",
len(recordsRaw))

    for i, rec := range recordsRaw {

        recBytes, _ := json.Marshal(rec)

        var cpi CPIRecord

        if err := json.Unmarshal(recBytes, &cpi); err != nil
{

            fmt.Printf("Error parseando registro %d: %v\n",
i, err)

            continue

```

```

    }

    if value, ok := cpi.GetNumericValue(); ok {

        fmt.Printf("Año: %s | CPI: %.2f\n", cpi.Date,
value)

    } else {

        fmt.Printf("Año: %s | CPI: Sin datos\n",
cpi.Date)

    }

}

}

}

}

```

He creado el archivo de prueba en Go:internal/experiments/worldbank_api_demo.go

Este archivo:

- Consulta el Índice de Precios al Consumidor (CPI) para Venezuela y, como segunda opción, Colombia usando la API pública del World Bank.
- Imprime en consola los valores de CPI por año.

¿Cómo probarlo?

1. Ve a la carpeta internal/experiments/.
1. Ejecuta cmd raíz del archivo

go run worldbank_api_demo.go

resultado cmd

PS

```
C:\Users\jhoan\Desktop\App_Simulador_Naci-n_Smithiana\cap1_division_del_trabajo\cap1_division_del_trabajo\internal\experiments>go run worldbank_api_demo.go
```

==== CPI para VE ===

Año: 2016 | CPI: 2740.27

Año: 2015 | CPI: 772.02

==== CPI para CO ===

Año: 2024 | CPI: 196.29

Año: 2023 | CPI: 184.12

Año: 2021 | CPI: 149.56

Año: 2020 | CPI: 144.51

Año: 2019 | CPI: 140.95

Año: 2018 | CPI: 136.15

Año: 2017 | CPI: 131.88

Año: 2021 | CPI: 149.56

Año: 2020 | CPI: 144.51

Año: 2019 | CPI: 140.95

Año: 2018 | CPI: 136.15

Año: 2017 | CPI: 131.88

```
Año: 2020 | CPI: 144.51
Año: 2019 | CPI: 140.95
Año: 2018 | CPI: 136.15
Año: 2017 | CPI: 131.88
Año: 2019 | CPI: 140.95
Año: 2018 | CPI: 136.15
Año: 2017 | CPI: 131.88
Año: 2018 | CPI: 136.15
Año: 2017 | CPI: 131.88
Año: 2017 | CPI: 131.88
Año: 2016 | CPI: 126.42
Año: 2015 | CPI: 117.59
Año: 2015 | CPI: 117.59
```

¿Qué logre?

- Consultate el Índice de Precios al Consumidor (CPI) real para Venezuela y Colombia.
- Los datos se muestran por año, tal como los entrega el World Bank.

¿Qué puedo hacer ahora con estos datos?

1. Simular inflación real en la app:
 - Puedo usar el CPI para ajustar el valor nominal de los productos año a año.

- Así, puedo mostrar cómo la inflación afecta el precio nominal, pero no el real (horas de trabajo).

1. Dashboard educativo:

- Mostrar gráficas de evolución del CPI junto a los precios de tus productos.
- Comparar el valor real (constante en horas de trabajo) vs. el valor nominal (fluctuante por inflación).

1. Extender a otros indicadores:

- Puedes consultar otros datos económicos (salarios, precios de commodities, etc.) usando la misma técnica.

Ahora voy a crear el sistema de precios dual para el Capítulo 5. Primero, voy a crear los modelos y la lógica para manejar productos con valores reales y nominales, integrando datos de inflación del World Bank API.

```
package models

import (
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "strconv"
    "time"
)
```



```

// Producto representa un bien con valor real (horas de trabajo)
y nominal (precio en moneda)

type Producto struct {

    ID            string          `json:"id"`

    Nombre        string          `json:"nombre"`

    HorasTrabajo  float64         `json:"horas_trabajo"`
// Valor real

    PrecioNominal float64         `json:"precio_nominal"`
// Valor nominal actual

    Moneda        string          `json:"moneda"`

    Pais          string          `json:"pais"`

    FechaCreacion time.Time       `json:"fecha_creacion"`

    HistorialPrecios []PrecioHistorico
`json:"historial_precios"`
}

// PrecioHistorico registra cambios en precios nominales

type PrecioHistorico struct {

    Fecha          time.Time `json:"fecha"`

    PrecioNominal  float64   `json:"precio_nominal"`

    FactorInflacion float64   `json:"factor_inflacion"`

```

```

    PrecioReal    float64    `json:"precio_real"`
}

// DatosInflacion representa datos de inflación del World Bank
type DatosInflacion struct {

    Pais          string      `json:"pais"`

    Indicador      string      `json:"indicador"`

    Datos          []DatoInflacion `json:"datos"`
}

type DatoInflacion struct {

    Ano    string    `json:"ano"`

    Valor  float64    `json:"valor"`
}

// Productos globales para el simulador
var Productos = map[string]*Producto{

    "trigo": {

        ID:          "trigo",

        Nombre:      "Trigo",

        HorasTrabajo: 2.0,
    },
}

```

```
PrecioNominal: 10.0,  
  
Moneda:      "USD",  
  
Pais:        "Venezuela",  
  
FechaCreacion: time.Now(),  
  
},  
  
"herramientas": {  
  
    ID:        "herramientas",  
  
    Nombre:     "Herramientas",  
  
    HorasTrabajo: 8.0,  
  
    PrecioNominal: 40.0,  
  
    Moneda:      "USD",  
  
    Pais:        "Venezuela",  
  
    FechaCreacion: time.Now(),  
  
},  
  
"ropa": {  
  
    ID:        "ropa",  
  
    Nombre:     "Ropa",  
  
    HorasTrabajo: 4.0,  
  
    PrecioNominal: 20.0,  
  
    Moneda:      "USD",  
  
    Pais:        "Venezuela",
```

```

        FechaCreacion: time.Now(),
    },

    "vivienda": {

        ID:            "vivienda",

        Nombre:        "Vivienda",

        HorasTrabajo:  100.0,

        PrecioNominal: 500.0,

        Moneda:        "USD",

        Pais:          "Venezuela",

        FechaCreacion: time.Now(),

    },

}

// ObtenerDatosInflacionWorldBank obtiene datos de inflación del
World Bank API

func ObtenerDatosInflacionWorldBank(pais, indicador string)
(*DatosInflacion, error) {

    // URL del World Bank API para CPI (Consumer Price Index)

    url :=
fmt.Sprintf("http://api.worldbank.org/v2/country/%s/indicator/%s
?format=json&per_page=20", pais, indicador)

```

```
fmt.Printf("Consultando API para %s: %s\n", pais, url)

resp, err := http.Get(url)

if err != nil {

    return nil, fmt.Errorf("error al obtener datos de
inflación: %v", err)

}

defer resp.Body.Close()

body, err := io.ReadAll(resp.Body)

if err != nil {

    return nil, fmt.Errorf("error al leer respuesta: %v",
err)

}

// Parsear respuesta del World Bank API
var response []interface{}

if err := json.Unmarshal(body, &response); err != nil {

    return nil, fmt.Errorf("error al parsear JSON: %v", err)

}
```

```
if len(response) < 2 {
    return nil, fmt.Errorf("respuesta inesperada del API")
}

// Extraer datos del segundo elemento del array
datosRaw, ok := response[1].([]interface{})

if !ok {
    return nil, fmt.Errorf("formato de datos inesperado")
}

fmt.Printf("Encontrados %d registros para %s\n",
len(datosRaw), pais)

var datos []DatoInflacion

for i, datoRaw := range datosRaw {
    datoMap, ok := datoRaw.(map[string]interface{})

    if !ok {
        fmt.Printf("Registro %d no es un mapa válido\n", i)
        continue
    }
}
```

```

// Extraer año y valor

ano, ok1 := datoMap["date"].(string)

if !ok1 {

    fmt.Printf("Registro %d: año no encontrado\n", i)

    continue

}

// El valor puede venir como string, número o nulo
var valor float64

switch v := datoMap["value"].(type) {

case string:

    if v == "" {

        fmt.Printf("Registro %d: valor vacío\n", i)

        continue // Ignorar valores vacíos

    }

    parsed, err := strconv.ParseFloat(v, 64)

    if err != nil {

        fmt.Printf("Registro %d: error parseando valor '%s': %v\n", i, v, err)

        continue // Ignorar valores que no se pueden
parsear

```

```

        }

        valor = parsed

    case float64:

        valor = v

    case int:

        valor = float64(v)

    case nil:

        fmt.Printf("Registro %d: valor nulo\n", i)

        continue // Ignorar valores nulos

    default:

        fmt.Printf("Registro %d: tipo de valor inesperado: %T\n", i, v)

        continue // Ignorar otros tipos

    }

    fmt.Printf("Registro %d: %s = %.2f\n", i, ano, valor)

    datos = append(datos, DatoInflacion{

        Ano:    ano,

        Valor:  valor,

    })

}

```



```
    fmt.Printf("Total de datos válidos para %s: %d\n", pais,
len(datos))

    return &DatosInflacion{

        Pais:      pais,

        Indicador: indicador,

        Datos:      datos,

    }, nil
}

// CalcularFactorInflacion calcula el factor de inflación
acumulada

func CalcularFactorInflacion(datos *DatosInflacion, anoBase
string) (float64, error) {

    if len(datos.Datos) == 0 {

        return 1.0, nil

    }

    // Encontrar el valor base

    var valorBase float64

    encontrado := false
```

```
for _, dato := range datos.Datos {  
    if dato.Ano == anoBase {  
        valorBase = dato.Valor  
        encontrado = true  
        break  
    }  
}  
  
if !encontrado || valorBase == 0 {  
    // Usar el valor más antiguo disponible como base  
    valorBase = datos.Datos[len(datos.Datos)-1].Valor  
}  
  
// Calcular factor de inflación (valor actual / valor base)  
valorActual := datos.Datos[0].Valor  
factor := valorActual / valorBase  
  
return factor, nil  
}
```

```
// ActualizarPrecioNominal actualiza el precio nominal basado en
inflación

func (p *Producto) ActualizarPrecioNominal(factorInflacion
float64) {

    // El precio real se mantiene constante (horas de trabajo)

    // El precio nominal se ajusta por inflación

    precioReal := p.HorasTrabajo * 5.0 // 5 USD por hora de
trabajo como base

    // Nuevo precio nominal = precio real * factor de inflación

    nuevoPrecio := precioReal * factorInflacion

    // Registrar en historial

    historial := PrecioHistorico{

        Fecha:          time.Now(),

        PrecioNominal:   nuevoPrecio,

        FactorInflacion: factorInflacion,

        PrecioReal:      precioReal,

    }

    p.HistorialPrecios = append(p.HistorialPrecios, historial)

    p.PrecioNominal = nuevoPrecio
}
```

```
}

// ObtenerPrecioReal devuelve el precio real en horas de trabajo
func (p *Producto) ObtenerPrecioReal() float64 {
    return p.HorasTrabajo
}

// ObtenerPrecioNominal devuelve el precio nominal actual
func (p *Producto) ObtenerPrecioNominal() float64 {
    return p.PrecioNominal
}

// CalcularPoderAdquisitivo calcula cuántas unidades se pueden
comprar con cierta cantidad de dinero
func (p *Producto) CalcularPoderAdquisitivo(cantidadDinero
float64) float64 {
    if p.PrecioNominal <= 0 {
        return 0
    }
    return cantidadDinero / p.PrecioNominal
}
```

```

// CompararPreciosRealVsNominal compara precios reales y
nominales

func CompararPreciosRealVsNominal() map[string]interface{} {

    resultado := make(map[string]interface{})

    for id, producto := range Productos {

        precioReal := producto.ObtenerPrecioReal()

        precioNominal := producto.ObtenerPrecioNominal()

        resultado[id] = map[string]interface{}{

            "nombre":          producto.Nombre,

            "precio_real":      precioReal,

            "precio_nominal":   precioNominal,

            "unidad_real":      "horas de trabajo",

            "unidad_nominal":   producto.Moneda,

            "relacion":         precioNominal / precioReal,

        }

    }

    return resultado
}

```

```

}

// ObtenerEstadisticasInflacion obtiene estadísticas de
inflación

func ObtenerEstadisticasInflacion() map[string]interface{} {

    // Obtener datos de inflación para Venezuela

    datosVenezuela, err := ObtenerDatosInflacionWorldBank("VE",
"FP.CPI.TOTL")

    if err != nil {

        return map[string]interface{}{

            "error": "No se pudieron obtener datos de inflación
de Venezuela: " + err.Error(),

        }

    }

    // Obtener datos de inflación para Colombia

    datosColombia, err2 := ObtenerDatosInflacionWorldBank("CO",
"FP.CPI.TOTL")

    if err2 != nil {

        return map[string]interface{}{

            "error": "No se pudieron obtener datos de inflación
de Colombia: " + err2.Error(),

        }

    }

}

```

```
}

// Verificar que hay datos disponibles

if len(datosVenezuela.Datos) == 0 ||
len(datosColombia.Datos) == 0 {

    return map[string]interface{}{

        "error": "No hay datos de inflación disponibles",

    }

}

// Calcular factores de inflación usando años disponibles

anoBaseVenezuela := "2015" // Usar 2015 como base para
Venezuela

if len(datosVenezuela.Datos) > 0 {

    anoBaseVenezuela =
datosVenezuela.Datos[len(datosVenezuela.Datos)-1].Ano // Usar el
año más antiguo disponible

}

anoBaseColombia := "2015" // Usar 2015 como base para
Colombia

if len(datosColombia.Datos) > 0 {
```

```

        anoBaseColombia =
datosColombia.Datos[len(datosColombia.Datos)-1].Ano // Usar el
año más antiguo disponible

    }

    factorVenezuela, _ :=
CalcularFactorInflacion(datosVenezuela, anoBaseVenezuela)

    factorColombia, _ := CalcularFactorInflacion(datosColombia,
anoBaseColombia)

    // Preparar datos recientes (máximo 5 años)

    datosRecientesVenezuela := datosVenezuela.Datos

    if len(datosRecientesVenezuela) > 5 {

        datosRecientesVenezuela = datosRecientesVenezuela[:5]

    }

    datosRecientesColombia := datosColombia.Datos

    if len(datosRecientesColombia) > 5 {

        datosRecientesColombia = datosRecientesColombia[:5]

    }

    return map[string]interface{}{

```



```

    "venezuela": map[string]interface{}{

        "factor_inflacion": factorVenezuela,

        "datos_recientes":  datosRecientesVenezuela,

        "ano_base":          anoBaseVenezuela,

    },

    "colombia": map[string]interface{}{

        "factor_inflacion": factorColombia,

        "datos_recientes":  datosRecientesColombia,

        "ano_base":          anoBaseColombia,

    },

    "comparacion": map[string]interface{}{

        "diferencia_factores": factorVenezuela -
factorColombia,

        "ratio_factores":      factorVenezuela /
factorColombia,

    },

    }

}

```

Ahora voy a agregar las rutas para el Capítulo 5 en el archivo [main.go](#):

```

"cap1_division_del_trabajo/internal/models"

```

Ahora voy a agregar la ruta para la página del Capítulo 5 y los endpoints correspondientes:

```
// Ruta para la página de precios reales vs nominales
router.GET("/precios", func(c *gin.Context) {
    c.HTML(http.StatusOK, "precios.html", gin.H{
        "title": "Precios Reales vs Nominales - Capítulo 5",
    })
})
```

Ahora voy a agregar los endpoints para el sistema de precios dual al final del archivo main.go:

```
// ===== ENDPOINTS DEL SISTEMA DE PRECIOS DUAL (CAPÍTULO 5)
=====

// Endpoint para obtener todos los productos con precios
reales y nominales
router.GET("/api/precios/productos", func(c *gin.Context) {
    productos := models.CompararPreciosRealVsNominal()
    c.JSON(http.StatusOK, gin.H{
        "success": true,
        "productos": productos,
    })
})
```

```
// Endpoint para obtener datos de inflación del World Bank
router.GET("/api/precios/inflacion", func(c *gin.Context) {

    estadisticas := models.ObtenerEstadisticasInflacion()

    c.JSON(http.StatusOK, gin.H{

        "success":      true,

        "estadisticas": estadisticas,

    })

})

// Endpoint para actualizar precios nominales basado en
inflación

router.GET("/api/precios/actualizar/:pais", func(c
*gin.Context) {

    pais := c.Param("pais")

    // Obtener datos de inflación

    datosInflacion, err :=
models.ObtenerDatosInflacionWorldBank(pais, "FP.CPI.TOTL")

    if err != nil {

        c.JSON(http.StatusInternalServerError, gin.H{

            "success": false,
```

```

        "error": "Error al obtener datos de inflación:
" + err.Error(),

    })

    return

}

// Calcular factor de inflación

factorInflacion, err :=
models.CalcularFactorInflacion(datosInflacion, "2010")

if err != nil {

    c.JSON(http.StatusInternalServerError, gin.H{

        "success": false,

        "error": "Error al calcular factor de
inflación: " + err.Error(),

    })

    return

}

// Actualizar precios de todos los productos

for _, producto := range models.Productos {

    if producto.Pais == pais {

```

```

producto.ActualizarPrecioNominal(factorInflacion)

    }

}

c.JSON(http.StatusOK, gin.H{

    "success":                true,

    "pais":                   pais,

    "factor_inflacion":       factorInflacion,

    "productos_actualizados": len(models.Productos),

})

})

// Endpoint para calcular poder adquisitivo

router.GET("/api/precios/poder-adquisitivo/:producto/:cantidad",
func(c *gin.Context) {

    productoID := c.Param("producto")

    cantidadStr := c.Param("cantidad")

    cantidad, err := strconv.ParseFloat(cantidadStr, 64)

    if err != nil {

```

```

        c.JSON(http.StatusBadRequest, gin.H{
            "success": false,
            "error":   "Cantidad inválida",
        })

        return
    }

    producto, existe := models.Productos[productoID]

    if !existe {
        c.JSON(http.StatusNotFound, gin.H{
            "success": false,
            "error":   "Producto no encontrado",
        })

        return
    }

    poderAdquisitivo :=
producto.CalcularPoderAdquisitivo(cantidad)

    c.JSON(http.StatusOK, gin.H{
        "success": true,
    })

```

```

        "producto":      producto.Nombre,

        "cantidad_dinero": cantidad,

        "moneda":        producto.Moneda,

        "poder_adquisitivo": poderAdquisitivo,

        "unidades":      poderAdquisitivo,

    })

})

// Endpoint para obtener historial de precios de un producto

router.GET("/api/precios/historial/:producto", func(c
*gin.Context) {

    productoID := c.Param("producto")

    producto, existe := models.Productos[productoID]

    if !existe {

        c.JSON(http.StatusNotFound, gin.H{

            "success": false,

            "error":    "Producto no encontrado",

        })

        return

    }

```

```
historial := producto.HistorialPrecios

if historial == nil {

    historial = []models.PrecioHistorico{}

}

c.JSON(http.StatusOK, gin.H{

    "success":    true,

    "producto":   producto.Nombre,

    "historial":  historial,

})

})
```

Crearemos el html

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

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



```
<title>{{.title}}</title>

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/boot
strap.min.css" rel="stylesheet">

<link
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/
css/all.min.css" rel="stylesheet">

<link href="/assets/css/precios.css" rel="stylesheet">

<script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<style>

    .hero-section {

        background: linear-gradient(135deg, #667eea 0%,
#764ba2 100%);

        color: white;

        padding: 4rem 0;

    }

    .theory-card {

        background: #f8f9fa;

        border-left: 4px solid #667eea;

        padding: 2rem;

        margin: 2rem 0;
```

```
}

.price-comparison {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    padding: 2rem;
    margin: 2rem 0;
}

.inflation-card {
    background: linear-gradient(135deg, #ff6b6b 0%,
#ee5a24 100%);
    color: white;
    border-radius: 10px;
    padding: 2rem;
    margin: 2rem 0;
}

.calculator-card {
```

```
        background: linear-gradient(135deg, #4ecdc4 0%,
#44a08d 100%);

        color: white;

        border-radius: 10px;

        padding: 2rem;

        margin: 2rem 0;
    }

    .nav-tabs .nav-link {

        color: #667eea;

        border: none;

        border-bottom: 2px solid transparent;
    }

    .nav-tabs .nav-link.active {

        color: #667eea;

        border-bottom: 2px solid #667eea;

        background: none;
    }

    .table th {
```

```
background: #667eea;

color: white;

}
```

```
.btn-primary {

background: #667eea;

border: none;

}
```

```
.btn-primary:hover {

background: #5a6fd8;

}
```

```
.loading {

display: none;

}
```

```
.chart-container {

position: relative;

height: 400px;

margin: 2rem 0;
```

```

    }

</style>

</head>

<body>

    <!-- Hero Section -->

    <div class="hero-section">

        <div class="container">

            <div class="row">

                <div class="col-12 text-center">

                    <h1 class="display-4 mb-4">

                        <i class="fas fa-balance-scale
me-3"></i>

                        Precios Reales vs Nominales

                    </h1>

                    <p class="lead">Capítulo 5: Del precio real
y nominal de las mercancías</p>

                    <p>Explora cómo la inflación afecta los
precios nominales mientras los valores reales permanecen
constantes</p>

                </div>

            </div>

        </div>

    </div>

</div>

```

```

</div>

<div class="container mt-5">

  <!-- Teoría -->

  <div class="theory-card">

    <h2><i class="fas fa-book me-2"></i>Teoría del
Valor-Trabajo</h2>

    <div class="row">

      <div class="col-md-6">

        <h4>Valor Real</h4>

        <p>La cantidad de trabajo requerido para
producir un bien. Este valor permanece constante
independientemente de las fluctuaciones monetarias.</p>

        <ul>

          <li>Se mide en horas de trabajo</li>

          <li>No cambia con la inflación</li>

          <li>Representa el verdadero costo de
producción</li>

        </ul>

      </div>

      <div class="col-md-6">

        <h4>Valor Nominal</h4>

```

```
<p>La cantidad de dinero que representa el
valor de un bien. Este valor puede cambiar debido a la inflación
o deflación.</p>
```

```
<ul>

  <li>Se mide en unidades monetarias</li>

  <li>Cambia con la inflación</li>

  <li>Puede distorsionar la percepción del
valor real</li>
```

```
</ul>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- Navegación por pestañas -->
```

```
<ul class="nav nav-tabs" id="preciosTabs"
role="tablist">
```

```
<li class="nav-item" role="presentation">
```

```
<button class="nav-link active"
id="productos-tab" data-bs-toggle="tab"
data-bs-target="#productos" type="button" role="tab">
```

```
<i class="fas fa-boxes me-2"></i>Productos
```

```
</button>
```

```
</li>
```

```

        <li class="nav-item" role="presentation">

            <button class="nav-link" id="inflacion-tab"
data-bs-toggle="tab" data-bs-target="#inflacion" type="button"
role="tab">

                <i class="fas fa-chart-line
me-2"></i>Inflación

            </button>

        </li>

        <li class="nav-item" role="presentation">

            <button class="nav-link" id="calculadora-tab"
data-bs-toggle="tab" data-bs-target="#calculadora" type="button"
role="tab">

                <i class="fas fa-calculator
me-2"></i>Calculadora

            </button>

        </li>

        <li class="nav-item" role="presentation">

            <button class="nav-link" id="historial-tab"
data-bs-toggle="tab" data-bs-target="#historial" type="button"
role="tab">

                <i class="fas fa-history me-2"></i>Historial

            </button>

        </li>

    </ul>

```



```

<div class="tab-content" id="preciosTabsContent">

    <!-- Pestaña de Productos -->

    <div class="tab-pane fade show active"
id="productos" role="tabpanel">

        <div class="price-comparison">

            <h3><i class="fas fa-table
me-2"></i>Comparación de Precios Reales vs Nominales</h3>

            <div class="loading" id="loadingProductos">

                <div class="text-center">

                    <div class="spinner-border
text-primary" role="status">

                        <span
class="visually-hidden">Cargando...</span>

                    </div>

                </div>

            </div>

            <div id="productosTable"></div>

        </div>

    </div>

    <!-- Pestaña de Inflación -->

```

```

        <div class="tab-pane fade" id="inflacion"
role="tabpanel">

        <div class="inflation-card">

            <h3><i class="fas fa-chart-line
me-2"></i>Datos de Inflación del World Bank</h3>

            <div class="row">

                <div class="col-md-6">

                    <h4>Venezuela</h4>

                    <div id="venezuelaStats"></div>

                </div>

                <div class="col-md-6">

                    <h4>Colombia</h4>

                    <div id="colombiaStats"></div>

                </div>

            </div>

            <div class="chart-container">

                <canvas id="inflationChart"></canvas>

            </div>

            <div class="mt-3">

                <button class="btn btn-light"
onclick="actualizarPrecios('VE') ">

```

```

                <i class="fas fa-sync-alt
me-2"></i>Actualizar Precios Venezuela

            </button>

            <button class="btn btn-light ms-2"
onclick="actualizarPrecios('CO') ">

                <i class="fas fa-sync-alt
me-2"></i>Actualizar Precios Colombia

            </button>

        </div>

    </div>

</div>

<!-- Pestaña de Calculadora -->

<div class="tab-pane fade" id="calculadora"
role="tabpanel">

    <div class="calculator-card">

        <h3><i class="fas fa-calculator
me-2"></i>Calculadora de Poder Adquisitivo</h3>

        <div class="row">

            <div class="col-md-6">

                <div class="mb-3">

                    <label for="productoSelect"
class="form-label">Producto:</label>

```

```

        <select class="form-select"
id="productoSelect">

        <option
value="trigo">Trigo</option>

        <option
value="herramientas">Herramientas</option>

        <option
value="ropa">Ropa</option>

        <option
value="vivienda">Vivienda</option>

        </select>

    </div>

    <div class="mb-3">

        <label for="cantidadDinero"
class="form-label">Cantidad de Dinero:</label>

        <input type="number"
class="form-control" id="cantidadDinero" value="100" min="0"
step="0.01">

    </div>

    <button class="btn btn-light"
onclick="calcularPoderAdquisitivo()">

        <i class="fas fa-calculator
me-2"></i>Calcular

    </button>

```

```

        </div>

        <div class="col-md-6">

            <div
id="resultadoCalculadora"></div>

            </div>

        </div>

    </div>

<!-- Pestaña de Historial -->

    <div class="tab-pane fade" id="historial"
role="tabpanel">

        <div class="price-comparison">

            <h3><i class="fas fa-history
me-2"></i>Historial de Precios</h3>

            <div class="mb-3">

                <label for="productoHistorial"
class="form-label">Seleccionar Producto:</label>

                <select class="form-select"
id="productoHistorial" onchange="cargarHistorial()">

                    <option value="trigo">Trigo</option>

                    <option
value="herramientas">Herramientas</option>

```

```

        <option value="ropa">Ropa</option>

        <option
value="vivienda">Vivienda</option>

        </select>

    </div>

    <div id="historialTable"></div>

</div>

</div>

</div>

</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootst
rap.bundle.min.js"></script>

<script>

    let inflationChart;

    // Cargar datos al iniciar

    document.addEventListener('DOMContentLoaded', function()
{

        cargarProductos();

        cargarDatosInflacion();

```

```
});

// Función para cargar productos

async function cargarProductos() {

document.getElementById('loadingProductos').style.display =
'block';

    try {

        const response = await
fetch('/api/precios/productos');

        const data = await response.json();

        if (data.success) {

            mostrarProductos(data.productos);

        }

    } catch (error) {

        console.error('Error al cargar productos:',
error);

    } finally {

document.getElementById('loadingProductos').style.display =
'none';

    }

}
```

```

    }

    // Función para mostrar productos en tabla

    function mostrarProductos(productos) {

        const container =
document.getElementById('productosTable');

        let html = `

            <table class="table table-striped">

                <thead>

                    <tr>

                        <th>Producto</th>

                        <th>Precio Real (horas)</th>

                        <th>Precio Nominal (USD)</th>

                        <th>Relación Nominal/Real</th>

                    </tr>

                </thead>

                <tbody>

                    `;

        for (const [id, producto] of
Object.entries(productos)) {

```



```

        html += `
            <tr>

<td><strong>${producto.nombre}</strong></td>

            <td>${producto.precio_real} horas</td>

<td>${producto.precio_nominal.toFixed(2)}</td>

            <td>${producto.relacion.toFixed(2)}</td>

        </tr>

    `;
}

html += '</tbody></table>';
container.innerHTML = html;
}

// Función para cargar datos de inflación
async function cargarDatosInflacion() {
    try {
        const response = await
fetch('/api/precios/inflacion');

        const data = await response.json();

```

```

        if (data.success) {

mostrarEstadisticasInflacion(data.estadisticas);

            crearGraficoInflacion(data.estadisticas);

        }

    } catch (error) {

        console.error('Error al cargar datos de
inflación:', error);

    }

}

// Función para mostrar estadísticas de inflación

function mostrarEstadisticasInflacion(estadisticas) {

    if (estadisticas.error) {

document.getElementById('venezuelaStats').innerHTML = '<p
class="text-warning">Error al cargar datos</p>';

document.getElementById('colombiaStats').innerHTML = '<p
class="text-warning">Error al cargar datos</p>';

        return;

    }

```

```

const venezuela = estadisticas.venezuela;

const colombia = estadisticas.colombia;

const comparacion = estadisticas.comparacion;

document.getElementById('venezuelaStats').innerHTML
= `
    <p><strong>Factor de Inflación:</strong>
    ${venezuela.factor_inflacion.toFixed(2)}</p>

    <p><strong>Inflación Acumulada:</strong>
    ${((venezuela.factor_inflacion - 1) * 100).toFixed(1)}%</p>
    `;

document.getElementById('colombiaStats').innerHTML =
`
    <p><strong>Factor de Inflación:</strong>
    ${colombia.factor_inflacion.toFixed(2)}</p>

    <p><strong>Inflación Acumulada:</strong>
    ${((colombia.factor_inflacion - 1) * 100).toFixed(1)}%</p>
    `;

}

// Función para crear gráfico de inflación

```

```
function crearGraficoInflacion(estadisticas) {

    const ctx =
document.getElementById('inflationChart').getContext('2d');

    if (inflationChart) {

        inflationChart.destroy();

    }

    const venezuelaData =
estadisticas.venezuela.datos_recientes.map(d => d.valor);

    const colombiaData =
estadisticas.colombia.datos_recientes.map(d => d.valor);

    const labels =
estadisticas.venezuela.datos_recientes.map(d => d.ano);

    inflationChart = new Chart(ctx, {

        type: 'line',

        data: {

            labels: labels,

            datasets: [{

                label: 'Venezuela',

                data: venezuelaData,
```

```

        borderColor: '#ff6b6b',
        backgroundColor: 'rgba(255, 107, 107,
0.1)',

        tension: 0.1
    }, {
        label: 'Colombia',
        data: colombiaData,
        borderColor: '#4ecdc4',
        backgroundColor: 'rgba(78, 205, 196,
0.1)',

        tension: 0.1
    }]
},
options: {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
        title: {
            display: true,
            text: 'Inflación Anual (%) - Últimos
5 años',
            color: 'white'

```

```
    },
    legend: {
      labels: {
        color: 'white'
      }
    },
  },
  scales: {
    y: {
      ticks: {
        color: 'white'
      },
      grid: {
        color: 'rgba(255, 255, 255,
0.1)'
      }
    },
    x: {
      ticks: {
        color: 'white'
      },
    },
  },
}
```

```

        grid: {
            color: 'rgba(255, 255, 255,
0.1)'
        }
    }
}

});
}

// Función para actualizar precios
async function actualizarPrecios(pais) {
    try {
        const response = await
fetch(`/api/precios/actualizar/${pais}`);

        const data = await response.json();

        if (data.success) {
            alert(`Precios actualizados para ${pais}.
Factor de inflación: ${data.factor_inflacion.toFixed(2)}`);
            cargarProductos();
        } else {

```

```
        alert('Error al actualizar precios: ' +
data.error);

    }

    } catch (error) {

        console.error('Error al actualizar precios:',
error);

        alert('Error al actualizar precios');

    }

}

// Función para calcular poder adquisitivo

async function calcularPoderAdquisitivo() {

    const producto =
document.getElementById('productoSelect').value;

    const cantidad =
document.getElementById('cantidadDinero').value;

    try {

        const response = await
fetch(`/api/precios/poder-adquisitivo/${producto}/${cantidad}`);

        const data = await response.json();

        if (data.success) {
```



```

document.getElementById('resultadoCalculadora').innerHTML = `
    <div class="alert alert-light">
        <h5>Resultado:</h5>
        <p><strong>Producto:</strong>
${data.producto}</p>
        <p><strong>Cantidad de
dinero:</strong> ${data.cantidad_dinero} ${data.moneda}</p>
        <p><strong>Poder
adquisitivo:</strong> ${data.poder_adquisitivo.toFixed(2)}
unidades</p>
    </div>
`;
    } else {
        alert('Error al calcular poder adquisitivo:
' + data.error);
    }
} catch (error) {
    console.error('Error al calcular poder
adquisitivo:', error);
    alert('Error al calcular poder adquisitivo');
}
}
}

```

```
// Función para cargar historial

async function cargarHistorial() {

    const producto =
document.getElementById('productoHistorial').value;

    try {

        const response = await
fetch(`/api/precios/historial/${producto}`);

        const data = await response.json();

        if (data.success) {

            mostrarHistorial(data.historial,
data.producto);

        } else {

            alert('Error al cargar historial: ' +
data.error);

        }

    } catch (error) {

        console.error('Error al cargar historial:',
error);

        alert('Error al cargar historial');

    }

}
```

```

    }

    // Función para mostrar historial

    function mostrarHistorial(historial, producto) {

        const container =
document.getElementById('historialTable');

        if (historial.length === 0) {

            container.innerHTML = '<p class="text-muted">No
hay historial disponible para este producto.</p>';

            return;

        }

        let html = `

            <h4>Historial de ${producto}</h4>

            <table class="table table-striped">

                <thead>

                    <tr>

                        <th>Fecha</th>

                        <th>Precio Nominal</th>

                        <th>Factor Inflación</th>

```

```

        <th>Precio Real</th>

        </tr>

    </thead>

    <tbody>

        `;

        historial.forEach(item => {

            const fecha = new
Date(item.fecha).toLocaleDateString();

            html += `

                <tr>

                    <td>${fecha}</td>

<td>${item.precio_nominal.toFixed(2)}</td>

<td>${item.factor_inflacion.toFixed(2)}</td>

                    <td>${item.precio_real.toFixed(2)}
horas</td>

                </tr>

            `;

        });

```

```
        html += '</tbody></table>';

        container.innerHTML = html;

    }

</script>

</body>

</html>
```

Creamos el css

```
/* Estilos específicos para el Capítulo 5 - Precios Reales vs
Nominales */

.hero-section {

    background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);

    color: white;

    padding: 4rem 0;

    text-align: center;

}

.hero-section h1 {

    font-size: 3rem;

    font-weight: 700;
```

```
margin-bottom: 1rem;
}

.hero-section .lead {
    font-size: 1.25rem;
    margin-bottom: 1rem;
}

.theory-card {
    background: #f8f9fa;
    border-left: 4px solid #667eea;
    padding: 2rem;
    margin: 2rem 0;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.theory-card h2 {
    color: #667eea;
    margin-bottom: 1.5rem;
}
```

```
.theory-card h4 {  
    color: #495057;  
    margin-bottom: 1rem;  
}  
  
.theory-card ul {  
    padding-left: 1.5rem;  
}  
  
.theory-card li {  
    margin-bottom: 0.5rem;  
    color: #6c757d;  
}  
  
.price-comparison {  
    background: white;  
    border-radius: 10px;  
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  
    padding: 2rem;  
    margin: 2rem 0;
```

```
}

.price-comparison h3 {
    color: #495057;
    margin-bottom: 1.5rem;
}

.inflation-card {
    background: linear-gradient(135deg, #044ff1 0%, #1b1918
100%);
    color: white;
    border-radius: 10px;
    padding: 2rem;
    margin: 2rem 0;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.inflation-card h3 {
    margin-bottom: 1.5rem;
}
```



```
.inflation-card h4 {  
  
    margin-bottom: 1rem;  
  
    font-weight: 600;  
  
}  
  
.calculator-card {  
  
    background: linear-gradient(135deg, #4ecdc4 0%, #44a08d  
100%);  
  
    color: white;  
  
    border-radius: 10px;  
  
    padding: 2rem;  
  
    margin: 2rem 0;  
  
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);  
  
}  
  
.calculator-card h3 {  
  
    margin-bottom: 1.5rem;  
  
}  
  
.nav-tabs {  
  
    border-bottom: 2px solid #dee2e6;
```

```
margin-bottom: 2rem;
}

.nav-tabs .nav-link {
    color: #667eea;

    border: none;

    border-bottom: 2px solid transparent;

    padding: 1rem 1.5rem;

    font-weight: 500;

    transition: all 0.3s ease;
}

.nav-tabs .nav-link:hover {
    color: #5a6fd8;

    border-bottom: 2px solid #5a6fd8;
}

.nav-tabs .nav-link.active {
    color: #667eea;

    border-bottom: 2px solid #667eea;

    background: none;
}
```

```
font-weight: 600;
}

.table {
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.table th {
    background: #667eea;
    color: white;
    border: none;
    padding: 1rem;
    font-weight: 600;
}

.table td {
    padding: 1rem;
    vertical-align: middle;
    border-color: #dee2e6;
```

```
}

.table-striped tbody tr:nth-of-type(odd) {
    background-color: rgba(102, 126, 234, 0.05);
}

.btn {
    border-radius: 6px;
    padding: 0.75rem 1.5rem;
    font-weight: 500;
    transition: all 0.3s ease;
}

.btn-primary {
    background: #667eea;
    border: none;
}

.btn-primary:hover {
    background: #5a6fd8;
    transform: translateY(-1px);
}
```

```
}

.btn-light {
    background: rgba(255, 255, 255, 0.9);
    border: none;
    color: #495057;
}

.btn-light:hover {
    background: white;
    transform: translateY(-1px);
}

.form-control, .form-select {
    border-radius: 6px;
    border: 2px solid #dee2e6;
    padding: 0.75rem;
    transition: border-color 0.3s ease;
}

.form-control:focus, .form-select:focus {
```

```
border-color: #667eea;

box-shadow: 0 0 0 0.2rem rgba(102, 126, 234, 0.25);
}

.form-label {

    font-weight: 600;

    color: #495057;

    margin-bottom: 0.5rem;
}

.loading {

    display: none;

    text-align: center;

    padding: 2rem;
}

.chart-container {

    position: relative;

    height: 400px;

    margin: 2rem 0;

    background: rgba(255, 255, 255, 0.1);
```

```
border-radius: 8px;

padding: 1rem;
}

.alert {

border-radius: 8px;

border: none;

padding: 1rem 1.5rem;
}

.alert-light {

background: rgba(255, 255, 255, 0.9);

color: #495057;
}

.text-warning {

color: #ffc107 !important;
}

.text-muted {

color: #6c757d !important;
}
```

```
}

/* Responsive design */
@media (max-width: 768px) {

    .hero-section h1 {

        font-size: 2rem;

    }

    .hero-section .lead {

        font-size: 1rem;

    }

    .nav-tabs .nav-link {

        padding: 0.75rem 1rem;

        font-size: 0.9rem;

    }

    .table th, .table td {

        padding: 0.75rem;

        font-size: 0.9rem;

    }

}
```



```
.chart-container {  
    height: 300px;  
}  
}  
  
@media (max-width: 576px) {  
    .hero-section {  
        padding: 2rem 0;  
    }  
  
    .hero-section h1 {  
        font-size: 1.75rem;  
    }  
  
    .theory-card, .price-comparison, .inflation-card,  
.calculator-card {  
        padding: 1.5rem;  
        margin: 1rem 0;  
    }  
}
```

```
.nav-tabs .nav-link {  
    padding: 0.5rem 0.75rem;  
    font-size: 0.8rem;  
}  
  
.btn {  
    padding: 0.5rem 1rem;  
    font-size: 0.9rem;  
}  
  
.chart-container {  
    height: 250px;  
}  
}  
  
/* Animaciones */  
@keyframes fadeIn {  
    from {  
        opacity: 0;  
        transform: translateY(20px);  
    }  
}
```

```
to {  
    opacity: 1;  
    transform: translateY(0);  
}  
}  
  
.tab-pane.fade.show {  
    animation: fadeIn 0.5s ease-out;  
}  
  
/* Estilos para iconos */  
.fas, .fa {  
    margin-right: 0.5rem;  
}  
  
/* Estilos para el spinner de carga */  
.spinner-border {  
    width: 3rem;  
    height: 3rem;  
}
```

```
/* Estilos para las tarjetas de estadísticas */
```

```
.stats-card {
```

```
    background: rgba(255, 255, 255, 0.1);
```

```
    border-radius: 8px;
```

```
    padding: 1.5rem;
```

```
    margin: 1rem 0;
```

```
    backdrop-filter: blur(10px);
```

```
}
```

```
.stats-card h5 {
```

```
    margin-bottom: 1rem;
```

```
    font-weight: 600;
```

```
}
```

```
.stats-card p {
```

```
    margin-bottom: 0.5rem;
```

```
    font-size: 0.9rem;
```

```
}
```

```
/* Estilos para el historial */
```

```
.historial-item {
```

```
background: #f8f9fa;

border-radius: 6px;

padding: 1rem;

margin: 0.5rem 0;

border-left: 3px solid #667eea;
}

.historial-item:hover {

background: #e9ecef;

transform: translateX(5px);

transition: all 0.3s ease;
}
```

Creando el [precio.js](#)

```
let inflationChart;

// Productos por país (ajusta según tus datos reales)

const productosPorPais = {

  VE: [

    { id: "trigo", nombre: "Trigo" },
```

```
        { id: "ropa", nombre: "Ropa" },
        { id: "vivienda", nombre: "Vivienda" }
    ],
    CO: [
        { id: "herramientas", nombre: "Herramientas" }
    ]
};

// Cargar datos al iniciar
document.addEventListener('DOMContentLoaded', function () {
    cargarProductos();
    cargarDatosInflacion();
    actualizarSelectProductos();
});

// Función para cargar productos
async function cargarProductos() {
    document.getElementById('loadingProductos').style.display =
    'block';

    try {
        const response = await fetch('/api/precios/productos');
```

```
const data = await response.json();

if (data.success) {

    mostrarProductos(data.productos);

}

} catch (error) {

    console.error('Error al cargar productos:', error);

} finally {

document.getElementById('loadingProductos').style.display =
'none';

}

}

// Función para mostrar productos en tabla

function mostrarProductos(productos) {

    const container = document.getElementById('productosTable');

    let html = `

        <table class="table table-striped">

            <thead>

                <tr>
```

```

        <th>Producto</th>

        <th>Precio Real  (horas)</th>

        <th>Precio Nominal  (USD)</th>

        <th>Relación Nominal/Real</th>

    </tr>

</thead>

<tbody>

`;

for (const [id, producto] of Object.entries(productos)) {
    html += `

        <tr>

            <td><strong>${producto.nombre}</strong></td>

            <td>${producto.precio_real} horas</td>

            <td>${producto.precio_nominal.toFixed(2)}</td>

            <td>${producto.relacion.toFixed(2)}</td>

        </tr>

    `;
}

html += '</tbody></table>';

```



```
        container.innerHTML = html;
    }

    // Función para cargar datos de inflación
    async function cargarDatosInflacion() {
        try {
            const response = await fetch('/api/precios/inflacion');
            const data = await response.json();

            if (data.success) {
                mostrarEstadisticasInflacion(data.estadisticas);
                crearGraficoInflacion(data.estadisticas);
            }
        } catch (error) {
            console.error('Error al cargar datos de inflación:',
error);
        }
    }

    // Función para mostrar estadísticas de inflación
    function mostrarEstadisticasInflacion(estadisticas) {
```

```
if (estadisticas.error) {

    document.getElementById('venezuelaStats').innerHTML =
    '<p class="text-warning">Error al cargar datos</p>';

    document.getElementById('colombiaStats').innerHTML = '<p
class="text-warning">Error al cargar datos</p>';

    return;

}

const venezuela = estadisticas.venezuela;

const colombia = estadisticas.colombia;

const comparacion = estadisticas.comparacion;

document.getElementById('venezuelaStats').innerHTML = `

    <p><strong>Factor de Inflación:</strong>
    ${venezuela.factor_inflacion.toFixed(2)}</p>

    <p><strong>Inflación Acumulada:</strong>
    ${((venezuela.factor_inflacion - 1) * 100).toFixed(1)}%</p>

`;

document.getElementById('colombiaStats').innerHTML = `

    <p><strong>Factor de Inflación:</strong>
    ${colombia.factor_inflacion.toFixed(2)}</p>
```

```

        <p><strong>Inflación Acumulada:</strong>
        ${{(colombia.factor_inflacion - 1) * 100).toFixed(1)}%</p>

        `;

    }

// Función para crear gráfico de inflación
function crearGraficoInflacion(estadisticas) {

    const ctx =
document.getElementById('inflationChart').getContext('2d');

    if (inflationChart) {

        inflationChart.destroy();

    }

    const venezuelaData =
estadisticas.venezuela.datos_recientes.map(d => d.valor);

    const colombiaData =
estadisticas.colombia.datos_recientes.map(d => d.valor);

    const labels = estadisticas.venezuela.datos_recientes.map(d
=> d.ano);

    inflationChart = new Chart(ctx, {

        type: 'line',

```

```
data: {  
  labels: labels,  
  datasets: [{  
    label: 'Venezuela',  
    data: venezuelaData,  
    borderColor: '#ff6b6b',  
    backgroundColor: 'rgba(255, 107, 107, 0.1)',  
    tension: 0.1  
  }, {  
    label: 'Colombia',  
    data: colombiaData,  
    borderColor: '#4ecdc4',  
    backgroundColor: 'rgba(78, 205, 196, 0.1)',  
    tension: 0.1  
  }]  
},  
options: {  
  responsive: true,  
  maintainAspectRatio: false,  
  plugins: {  
    title: {
```

```

        display: true,

        text: 'Inflación Anual (%) - Últimos 5
años',

        color: 'white'

    },

    legend: {

        labels: {

            color: 'white'

        }

    },

},

scales: {

    y: {

        ticks: {

            color: 'white'

        },

        grid: {

            color: 'rgba(255, 255, 255, 0.1)'

        }

    },

    x: {

```

```
        ticks: {
            color: 'white'
        },
        grid: {
            color: 'rgba(255, 255, 255, 0.1)'
        }
    }
}

});
}

// Función para actualizar precios
async function actualizarPrecios(pais) {
    try {
        const response = await
fetch(`/api/precios/actualizar/${pais}`);

        const data = await response.json();

        if (data.success) {
```

```

        alert(`Precios actualizados para ${pais}. Factor de
inflación: ${data.factor_inflacion.toFixed(2)}`);

        cargarProductos();

    } else {

        alert('Error al actualizar precios: ' + data.error);

    }

} catch (error) {

    console.error('Error al actualizar precios:', error);

    alert('Error al actualizar precios');

}

}

// Función para calcular poder adquisitivo
async function calcularPoderAdquisitivo() {

    const producto =
document.getElementById('productoSelect').value;

    const cantidad =
document.getElementById('cantidadDinero').value;

    try {

        const response = await
fetch(`/api/precios/poder-adquisitivo/${producto}/${cantidad}`);

```

```

const data = await response.json();

if (data.success) {

document.getElementById('resultadoCalculadora').innerHTML = `

    <div class="alert alert-light">

        <h5>Resultado:</h5>

        <p><strong>Producto:</strong>
${data.producto}</p>

        <p><strong>Cantidad de dinero:</strong>
${data.cantidad_dinero} ${data.moneda}</p>

        <p><strong>Poder adquisitivo:</strong>
${data.poder_adquisitivo.toFixed(2)} unidades</p>

    </div>

    `;

} else {

    alert('Error al calcular poder adquisitivo: ' +
data.error);

}

} catch (error) {

    console.error('Error al calcular poder adquisitivo:',
error);

    alert('Error al calcular poder adquisitivo');

```



```
    }  
  }  
  
  // Función para cargar historial  
  
  async function cargarHistorial() {  
  
    const producto =  
document.getElementById('productoHistorial').value;  
  
    if (!producto) return;  
  
    try {  
  
      const response = await  
fetch(`/api/precios/historial/${producto}`);  
  
      const data = await response.json();  
  
      if (data.success) {  
  
        mostrarHistorial(data.historial, data.producto);  
  
      } else {  
  
        alert('Error al cargar historial: ' + data.error);  
  
      }  
  
    } catch (error) {  
  
      console.error('Error al cargar historial:', error);  
  
      alert('Error al cargar historial');  
  
    }  
  }  
}
```

```

}

// Función para mostrar historial

function mostrarHistorial(historial, producto) {

    console.log("Historial recibido:", historial, "Producto:",
producto);

    if (!Array.isArray(historial)) historial = [];

    const container = document.getElementById('historialTable');

    if (historial.length === 0) {

        container.innerHTML = '<p class="text-muted">No hay
historial disponible para este producto.</p>';

        return;

    }

    let html = `

        <h4>Historial de ${producto}</h4>

        <table class="table table-striped">

            <thead>

                <tr>

                    <th>Fecha</th>

```

```

        <th>Precio Nominal</th>

        <th>Factor Inflación</th>

        <th>Precio Real</th>

    </tr>

</thead>

<tbody>

`;

historial.forEach(item => {

    const fecha = new Date(item.fecha).toLocaleDateString();

    html += `

        <tr>

            <td>${fecha}</td>

            <td>${item.precio_nominal.toFixed(2)}</td>

            <td>${item.factor_inflacion.toFixed(2)}</td>

            <td>${item.precio_real.toFixed(2)} horas</td>

        </tr>

    `;

});

html += '</tbody></table>';

```

```
        container.innerHTML = html;
    }

function actualizarSelectProductos() {

    const pais = document.getElementById('paisHistorial').value;

    const select = document.getElementById('productoHistorial');

    select.innerHTML = productosPorPais[pais].map(p => `<option
value="${p.id}">${p.nombre}</option>`).join('');

    cargarHistorial(); // Carga el historial del primer producto
del país seleccionado
}
```

Capítulo 6: De los componentes del precio de las mercancías

Conceptos Clave:

Estructura del precio:

Salarios (remuneración del trabajo)

Beneficios (remuneración del capital)

Rentas (remuneración de la tierra)

Precio natural vs. precio de mercado:

Precio natural = Costo de producción (salarios + beneficios + rentas)

Precio de mercado = Determinado por oferta y demanda

Aplicación en tu App:

Desglose de precios:

```
json

"productos": [

  {

    "nombre": "Pan",

    "precio_mercado": 10,

    "componentes": {

      "salarios": 4,      // 40%

      "beneficios": 3,   // 30%

      "rentas": 3        // 30%

    },

    "historial_precios": {

      "natural": [8, 8.5, 9],

      "mercado": [9, 9.5, 10]

    }

  }

]
```

Simulador de mercado:

```
javascript

class Mercado {

  constructor(producto) {

    this.producto = producto;

    this.oferta = 100;

    this.demanda = 100;

  }

  actualizarPrecio() {

    const desequilibrio = this.demanda / this.oferta;

    this.producto.precio_mercado = this.producto.precio_natural
* desequilibrio;

    return this.producto.precio_mercado;

  }

}
```

creando el data.json

Vamos a crear una estructura de data.json local con productos que incluya los componentes de precio del Capítulo 6, y luego crear una API para consumir esos datos.

```
{

  "usuarios": [
```

```
{
  "id": 1,
  "nombre": "Juan Pérez",
  "rol": "agricultor",
  "especializacion": "trigo",
  "productividad": 3.5,
  "saldo": 1500,
  "inventario": {
    "trigo": 100,
    "herramientas": 5,
    "dinero": 1500
  }
},
{
  "id": 2,
  "nombre": "María Gómez",
  "rol": "artesano",
  "especializacion": "herramientas",
  "productividad": 4.0,
  "saldo": 1200,
  "inventario": {
```

```
    "trigo": 20,  
    "herramientas": 30,  
    "dinero": 1200  
  },  
  },  
  {  
    "id": 3,  
    "nombre": "Carlos Ruiz",  
    "rol": "comerciante",  
    "especializacion": "transacciones",  
    "productividad": 2.5,  
    "saldo": 3000,  
    "inventario": {  
      "trigo": 50,  
      "herramientas": 15,  
      "dinero": 3000  
    }  
  }  
],  
"productos": [  
  {
```



```
"id": "trigo",

"nombre": "Trigo",

"categoria": "agricultura",

"precio_mercado": 250.00,

"precio_natural": 220.00,

"componentes": {

    "salarios": 112.50,

    "beneficios": 75.00,

    "rentas": 62.50

},

"historial_precios": {

    "natural": [200.00, 210.00, 215.00, 218.00, 220.00],

    "mercado": [210.00, 225.00, 235.00, 245.00, 250.00]

},

"oferta": 1000,

"demanda": 1200,

"pais": "VE",

"unidad": "tonelada"

},

{

    "id": "maiz",
```

```
"nombre": "Maíz",

"categoria": "agricultura",

"precio_mercado": 180.00,

"precio_natural": 160.00,

"componentes": {

  "salarios": 81.00,

  "beneficios": 54.00,

  "rentas": 45.00

},

"historial_precios": {

  "natural": [150.00, 155.00, 158.00, 159.00, 160.00],

  "mercado": [160.00, 165.00, 170.00, 175.00, 180.00]

},

"oferta": 800,

"demanda": 900,

"pais": "VE",

"unidad": "tonelada"

},

{

  "id": "herramientas",

  "nombre": "Herramientas",
```

```
"categoria": "manufactura",

"precio_mercado": 45.00,

"precio_natural": 40.00,

"componentes": {

    "salarios": 20.25,

    "beneficios": 13.50,

    "rentas": 11.25

},

"historial_precios": {

    "natural": [38.00, 39.00, 39.50, 39.80, 40.00],

    "mercado": [40.00, 42.00, 43.00, 44.00, 45.00]

},

"oferta": 500,

"demanda": 600,

"pais": "VE",

"unidad": "unidad"

},

{

    "id": "pan",

    "nombre": "Pan",

    "categoria": "alimentos",
```

```
"precio_mercado": 2.50,  
"precio_natural": 2.20,  
"componentes": {  
  "salarios": 1.13,  
  "beneficios": 0.75,  
  "rentas": 0.63  
},  
"historial_precios": {  
  "natural": [2.00, 2.10, 2.15, 2.18, 2.20],  
  "mercado": [2.10, 2.20, 2.30, 2.40, 2.50]  
},  
"oferta": 2000,  
"demanda": 2500,  
"pais": "VE",  
"unidad": "kg"  
},  
{  
  "id": "ropa",  
  "nombre": "Ropa",  
  "categoria": "textiles",  
  "precio_mercado": 35.00,
```

```
"precio_natural": 30.00,

"componentes": {

    "salarios": 15.75,

    "beneficios": 10.50,

    "rentas": 8.75

},

"historial_precios": {

    "natural": [28.00, 29.00, 29.50, 29.80, 30.00],

    "mercado": [30.00, 32.00, 33.00, 34.00, 35.00]

},

"oferta": 300,

"demanda": 400,

"pais": "VE",

"unidad": "unidad"

},

{

    "id": "vivienda",

    "nombre": "Vivienda",

    "categoria": "construccion",

    "precio_mercado": 50000.00,

    "precio_natural": 45000.00,
```

```
"componentes": {  
  "salarios": 22500.00,  
  "beneficios": 15000.00,  
  "rentas": 12500.00  
},  
  
"historial_precios": {  
  "natural": [40000.00, 42000.00, 43000.00, 44000.00,  
45000.00],  
  "mercado": [42000.00, 44000.00, 46000.00, 48000.00,  
50000.00]  
},  
  
"oferta": 50,  
  
"demanda": 80,  
  
"pais": "VE",  
  
"unidad": "m2"  
},  
  
{  
  
  "id": "trigo_co",  
  
  "nombre": "Trigo",  
  
  "categoria": "agricultura",  
  
  "precio_mercado": 280.00,
```

```
"precio_natural": 250.00,

"componentes": {

  "salarios": 126.00,

  "beneficios": 84.00,

  "rentas": 70.00

},

"historial_precios": {

  "natural": [230.00, 240.00, 245.00, 248.00, 250.00],

  "mercado": [240.00, 250.00, 260.00, 270.00, 280.00]

},

"oferta": 1200,

"demanda": 1100,

"pais": "CO",

"unidad": "tonelada"

},

{

  "id": "cafe",

  "nombre": "Café",

  "categoria": "agricultura",

  "precio_mercado": 320.00,

  "precio_natural": 290.00,
```

```
"componentes": {
  "salarios": 144.00,
  "beneficios": 96.00,
  "rentas": 80.00
},
"historial_precios": {
  "natural": [270.00, 280.00, 285.00, 288.00, 290.00],
  "mercado": [280.00, 290.00, 300.00, 310.00, 320.00]
},
"oferta": 600,
"demanda": 700,
"pais": "CO",
"unidad": "tonelada"
},
{
  "id": "bananas",
  "nombre": "Bananas",
  "categoria": "agricultura",
  "precio_mercado": 1.20,
  "precio_natural": 1.00,
  "componentes": {
```



```
"salarios": 0.54,

"beneficios": 0.36,

"rentas": 0.30

},

"historial_precios": {

  "natural": [0.90, 0.95, 0.98, 0.99, 1.00],

  "mercado": [1.00, 1.05, 1.10, 1.15, 1.20]

},

"oferta": 5000,

"demanda": 6000,

"pais": "CO",

"unidad": "kg"

}

],

"mercados": [

  {

    "id": "mercado_ve",

    "nombre": "Mercado de Venezuela",

    "pais": "VE",

    "productos": ["trigo", "maiz", "herramientas", "pan",

"ropa", "vivienda"],
```

```
"configuracion": {
  "factor_oferta_demanda": 1.2,
  "volatilidad_precio": 0.15,
  "inflacion_anual": 0.25
},
{
  "id": "mercado_co",
  "nombre": "Mercado de Colombia",
  "pais": "CO",
  "productos": ["trigo_co", "cafe", "bananas"],
  "configuracion": {
    "factor_oferta_demanda": 1.1,
    "volatilidad_precio": 0.10,
    "inflacion_anual": 0.08
  }
},
"indicadores_economicos": {
  "venezuela": {
    "inflacion_anual": 25.0,
```

```
    "pib_per_capita": 3500,  
    "tasa_desempleo": 15.0,  
    "salario_minimo": 150  
  },  
  "colombia": {  
    "inflacion_anual": 8.0,  
    "pib_per_capita": 6500,  
    "tasa_desempleo": 10.0,  
    "salario_minimo": 300  
  }  
}  
}
```

creando cap6_handlers.go

voy a crear una API en Go para consumir estos datos. Primero, voy a revisar la estructura de handlers existente:

```
package handlers  
  
import (  
    "net/http"  
    "strconv"
```

```

    "github.com/gin-gonic/gin"
)

// Estructuras para el Capítulo 6
type Producto struct {
    ID            string    `json:"id"`
    Nombre        string    `json:"nombre"`
    Categoria     string    `json:"categoria"`
    PrecioMercado float64    `json:"precio_mercado"`
    PrecioNatural float64    `json:"precio_natural"`
    Componentes   ComponentesPrecio `json:"componentes"`
    HistorialPrecios HistorialPrecios `json:"historial_precios"`
    Oferta        int        `json:"oferta"`
    Demanda       int        `json:"demanda"`
    Pais          string    `json:"pais"`
    Unidad        string    `json:"unidad"`
}

type ComponentesPrecio struct {
    Salarios float64 `json:"salarios"`

```

```

    Beneficios float64 `json:"beneficios"`

    Rentas      float64 `json:"rentas"`
}

type HistorialPrecios struct {

    Natural []float64 `json:"natural"`

    Mercado []float64 `json:"mercado"`
}

type Mercado struct {

    ID          string      `json:"id"`
    Nombre      string      `json:"nombre"`
    Pais        string      `json:"pais"`
    Productos   []string    `json:"productos"`
    Configuracion ConfiguracionMercado `json:"configuracion"`
}

type ConfiguracionMercado struct {

    FactorOfertaDemanda float64 `json:"factor_oferta_demanda"`

    VolatilidadPrecio   float64 `json:"volatilidad_precio"`

    InflacionAnual      float64 `json:"inflacion_anual"`
}

```

```

}

type IndicadoresEconomicos struct {

    Venezuela IndicadorPais `json:"venezuela"`

    Colombia IndicadorPais `json:"colombia"`

}

type IndicadorPais struct {

    InflacionAnual float64 `json:"inflacion_anual"`

    PibPerCapita float64 `json:"pib_per_capita"`

    TasaDesempleo float64 `json:"tasa_desempleo"`

    SalarioMinimo float64 `json:"salario_minimo"`

}

type DataCap6 struct {

    Usuarios []interface{}
    `json:"usuarios"`

    Productos []Producto
    `json:"productos"`

    Mercados []Mercado
    `json:"mercados"`

```

```

    IndicadoresEconomicos IndicadoresEconomicos
`json:"indicadores_economicos"`
}

var dataCap6 DataCap6

// Función para cargar datos del Capítulo 6
func CargarDatosCap6() error {

    // En un entorno real, cargarías desde un archivo

    // Por ahora, usaremos datos hardcoded para demostración

    dataCap6 = DataCap6{

        Productos: []Producto{

            {

                ID:          "trigo",

                Nombre:      "Trigo",

                Categoria:    "agricultura",

                PrecioMercado: 250.00,

                PrecioNatural: 220.00,

                Componentes: ComponentesPrecio{

                    Salarios: 112.50,

                    Beneficios: 75.00,

```

```

        Rentas:      62.50,

    },

    HistorialPrecios: HistorialPrecios{

        Natural: []float64{200.00, 210.00, 215.00,
218.00, 220.00},

        Mercado: []float64{210.00, 225.00, 235.00,
245.00, 250.00},

    },

    Oferta:  1000,

    Demanda: 1200,

    Pais:    "VE",

    Unidad:  "tonelada",

},

{

    ID:      "maiz",

    Nombre:  "Maíz",

    Categoria:  "agricultura",

    PrecioMercado: 180.00,

    PrecioNatural: 160.00,

    Componentes: ComponentesPrecio{

        Salarios:  81.00,

```



```

        Beneficios: 54.00,

        Rentas:      45.00,

    },

    HistorialPrecios: HistorialPrecios{

        Natural: []float64{150.00, 155.00, 158.00,
159.00, 160.00},

        Mercado: []float64{160.00, 165.00, 170.00,
175.00, 180.00},

    },

    Oferta: 800,

    Demanda: 900,

    Pais:    "VE",

    Unidad:  "tonelada",

},

{

    ID:          "herramientas",

    Nombre:      "Herramientas",

    Categoria:   "manufactura",

    PrecioMercado: 45.00,

    PrecioNatural: 40.00,

    Componentes: ComponentesPrecio{

```

```
        Salarios:    20.25,

        Beneficios: 13.50,

        Rentas:     11.25,

    },

    HistorialPrecios: HistorialPrecios{

        Natural: []float64{38.00, 39.00, 39.50,
39.80, 40.00},

        Mercado: []float64{40.00, 42.00, 43.00,
44.00, 45.00},

    },

    Oferta:  500,

    Demanda: 600,

    Pais:    "VE",

    Unidad:  "unidad",

    },

},

Mercados: []Mercado{

    {

        ID:        "mercado_ve",

        Nombre:    "Mercado de Venezuela",

        Pais:      "VE",
```

```
        Productos: []string{"trigo", "maiz",  
"herramientas"},  
  
        Configuracion: ConfiguracionMercado{  
  
            FactorOfertaDemanda: 1.2,  
  
            VolatilidadPrecio:    0.15,  
  
            InflacionAnual:      0.25,  
  
        },  
  
    },  
  
    },  
  
    IndicadoresEconomicos: IndicadoresEconomicos{  
  
        Venezuela: IndicadorPais{  
  
            InflacionAnual: 25.0,  
  
            PibPerCapita:   3500,  
  
            TasaDesempleo:  15.0,  
  
            SalarioMinimo:  150,  
  
        },  
  
        Colombia: IndicadorPais{  
  
            InflacionAnual: 8.0,  
  
            PibPerCapita:   6500,  
  
            TasaDesempleo:  10.0,  
  
            SalarioMinimo:  300,
```

```

        },

    },

}

return nil
}

// Handler para obtener todos los productos
func GetProductos(c *gin.Context) {

    c.JSON(http.StatusOK, gin.H{

        "success":    true,

        "productos": dataCap6.Productos,

    })

}

// Handler para obtener un producto específico
func GetProducto(c *gin.Context) {

    productoID := c.Param("id")

    for _, producto := range dataCap6.Productos {

        if producto.ID == productoID {

            c.JSON(http.StatusOK, gin.H{

```

```

        "success": true,

        "producto": producto,

    })

    return

}

}

c.JSON(http.StatusNotFound, gin.H{

    "success": false,

    "error": "Producto no encontrado",

})

}

// Handler para obtener productos por país
func GetProductosPorPais(c *gin.Context) {

    pais := c.Param("pais")

    var productosPais []Producto

    for _, producto := range dataCap6.Productos {

        if producto.Pais == pais {

            productosPais = append(productosPais, producto)

```

```

    }

}

c.JSON(http.StatusOK, gin.H{
    "success":    true,
    "productos":  productosPais,
    "pais":       pais,
})
}

// Handler para simular cambio en oferta/demanda
func SimularMercado(c *gin.Context) {
    productoID := c.Param("id")

    // Buscar el producto
    var producto *Producto

    for i := range dataCap6.Productos {
        if dataCap6.Productos[i].ID == productoID {
            producto = &dataCap6.Productos[i]
            break
        }
    }
}

```

```
}

if producto == nil {

    c.JSON(http.StatusNotFound, gin.H{

        "success": false,

        "error": "Producto no encontrado",

    })

    return

}

// Obtener parámetros de la query

ofertaStr := c.Query("oferta")

demandaStr := c.Query("demanda")

var nuevaOferta, nuevaDemanda int

var err error

if ofertaStr != "" {

    nuevaOferta, err = strconv.Atoi(ofertaStr)

    if err != nil {

        c.JSON(http.StatusBadRequest, gin.H{
```

```
        "success": false,
        "error": "Oferta debe ser un número válido",
    })
    return
}

} else {
    nuevaOferta = producto.Oferta
}

if demandaStr != "" {
    nuevaDemanda, err = strconv.Atoi(demandaStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "success": false,
            "error": "Demanda debe ser un número válido",
        })
        return
    }
} else {
    nuevaDemanda = producto.Demanda
}
```



```
// Calcular nuevo precio de mercado según Adam Smith

precioAnterior := producto.PrecioMercado

producto.Oferta = nuevaOferta

producto.Demanda = nuevaDemanda


// Fórmula simple: precio = precio_natural * (demanda /
oferta)

if nuevaOferta > 0 {

    factorMercado := float64(nuevaDemanda) /
float64(nuevaOferta)

    producto.PrecioMercado = producto.PrecioNatural *
factorMercado

}


// Actualizar componentes proporcionalmente

producto.Componentes.Salarios = producto.PrecioMercado *
0.45

producto.Componentes.Beneficios = producto.PrecioMercado *
0.30

producto.Componentes.Rentas = producto.PrecioMercado * 0.25


// Agregar al historial
```

```
    producto.HistorialPrecios.Natural =
append(producto.HistorialPrecios.Natural,
producto.PrecioNatural)

    producto.HistorialPrecios.Mercado =
append(producto.HistorialPrecios.Mercado,
producto.PrecioMercado)

    // Limitar historial a 10 elementos

    if len(producto.HistorialPrecios.Natural) > 10 {

        producto.HistorialPrecios.Natural =
producto.HistorialPrecios.Natural[1:]

        producto.HistorialPrecios.Mercado =
producto.HistorialPrecios.Mercado[1:]

    }

    c.JSON(http.StatusOK, gin.H{

        "success": true,

        "producto": producto,

        "cambios": gin.H{

            "precio_anterior": precioAnterior,

            "precio_nuevo": producto.PrecioMercado,

            "oferta_anterior": producto.Oferta,

            "oferta_nueva": nuevaOferta,
```

```

        "demanda_anterior": producto.Demanda,
        "demanda_nueva":     nuevaDemanda,
    },
    })
}

// Handler para obtener indicadores económicos
func GetIndicadoresEconomicos(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
        "success":     true,
        "indicadores": dataCap6.IndicadoresEconomicos,
    })
}

// Handler para obtener mercados
func GetMercados(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{
        "success":     true,
        "mercados":    dataCap6.Mercados,
    })
}

```

```
// Handler para análisis de componentes de precio

func AnalizarComponentes(c *gin.Context) {

    productoID := c.Param("id")

    var producto *Producto

    for i := range dataCap6.Productos {

        if dataCap6.Productos[i].ID == productoID {

            producto = &dataCap6.Productos[i]

            break

        }

    }

    if producto == nil {

        c.JSON(http.StatusNotFound, gin.H{

            "success": false,

            "error": "Producto no encontrado",

        })

        return

    }

}
```

```

// Calcular porcentajes

total := producto.PrecioMercado

porcentajeSalarios := (producto.Componentes.Salarios /
total) * 100

porcentajeBeneficios := (producto.Componentes.Beneficios /
total) * 100

porcentajeRentas := (producto.Componentes.Rentas / total) *
100

 analisis := gin.H{

    "producto": producto,

    "analisis": gin.H{

        "porcentajes": gin.H{

            "salarios":    porcentajeSalarios,

            "beneficios": porcentajeBeneficios,

            "rentas":      porcentajeRentas,

        },

        "explicacion": gin.H{

            "salarios":    "Remuneración del trabajo (según
Adam Smith)",

            "beneficios": "Remuneración del capital
invertido",

```

```

        "rentas":      "Remuneración de la tierra y
recursos naturales",

        },

        "precio_natural": producto.PrecioNatural,

        "precio_mercado": producto.PrecioMercado,

        "diferencia":   producto.PrecioMercado -
producto.PrecioNatural,

        },

    }

    c.JSON(http.StatusOK, gin.H{

        "success": true,

        "data":     analisis,

    })

}

```

creando el html para el cap6

voy a crear un archivo HTML para probar la API del Capítulo 6:

```

<!DOCTYPE html>

<html lang="es">

<head>

```

```
<meta charset="UTF-8">

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

<title>Simulador de Precios - Capítulo 6 (Adam
Smith)</title>

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/boot
strap.min.css" rel="stylesheet">

<link
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/
css/all.min.css" rel="stylesheet">

<script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<link rel="stylesheet" href="/assets/css/cap6.css">

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

<header>

</header>

<body>

<nav class="navbar-white">

    <div class="logo">
```

```


<h1>LOGO</h1>

</div>

<ul>

  <li>

    <a href="/expansion">Cap Anterior</a>

  </li>

  <li>

    <a href="/dinero">Cap 4: El origen y uso del
dinero</a>

  </li>

  <li>

    <a href="/precios">Cap 5: Precios Reales vs
Nominales</a>

  </li>

  <li>

    <a href="/cap6">Cap 6: Componentes del precio</a>

  </li>

  <li>

    <a href="/dinero">Siguietes Cap</a>

  </li>


```



```
</ul>

<div class="hamburger">

    <span class="line"></span>

    <span class="line"></span>

    <span class="line"></span>

</div>

</nav>

<div class="menubar">

    <ul>

        <li>

            <a href="/expansion">Cap Anterior</a>

        </li>

        <li>

            <a href="/dinero">Cap 4: El origen y uso del
dinero</a>

        </li>

        <li>

            <a href="/precios">Cap 5: Precios Reales vs
Nominales</a>

        </li>

        <li>
```

```

        <a href="/cap6">Cap 6: Componentes del precio</a>

    </li>

    <li>

        <a href="/dinero">Siguietes Cap</a>

    </li>

</ul>

</div>

<div class="container-fluid py-4">

    <div class="row">

        <div class="col-12 text-center mb-4">

            <h1><i class="fas fa-chart-line"></i> Simulador
de Precios - Capítulo 6</h1>

            <p class="lead">"De los componentes del precio
de las mercancías" - Adam Smith</p>

        </div>

    </div>

    <!-- Selector de Producto -->

    <div class="row mb-4">

        <div class="col-md-6">

            <div class="card">

```

```

        <div class="card-header">

            <h5><i class="fas fa-box"></i>
Seleccionar Producto</h5>

        </div>

        <div class="card-body">

            <select id="productoSelect"
class="form-select mb-3" onchange="cargarProducto()">

                <option value="">Selecciona un
producto...</option>

            </select>

            <div id="loadingProducto"
class="text-center" style="display: none;">

                <div class="spinner-border"
role="status">

                    <span
class="visually-hidden">Cargando...</span>

                </div>

            </div>

        </div>

    </div>

</div>

<div class="col-md-6">

    <div class="card">

```

```

        <div class="card-header">

            <h5><i class="fas fa-globe"></i>
Indicadores Económicos</h5>

        </div>

        <div class="card-body"
id="indicadoresEconomicos">

            <p class="text-muted">Selecciona un
producto para ver los indicadores</p>

        </div>

    </div>

</div>

</div>

</div>

<!-- Información del Producto -->

<div class="row mb-4" id="infoProducto" style="display:
none;">

    <div class="col-12">

        <div class="card">

            <div class="card-header">

                <h5><i class="fas fa-info-circle"></i>
Información del Producto</h5>

            </div>

            <div class="card-body">

```

```

        <div class="row">

            <div class="col-md-6">

                <h6>Precios</h6>

                <p><strong>Precio
Natural:</strong> $<span id="precioNatural">0</span></p>

                <p><strong>Precio de
Mercado:</strong> $<span id="precioMercado">0</span></p>

                <p><strong>Diferencia:</strong>
$<span id="diferenciaPrecio">0</span></p>

            </div>

            <div class="col-md-6">

                <h6>Mercado</h6>

                <p><strong>Oferta:</strong>
<span id="oferta">0</span> unidades</p>

                <p><strong>Demanda:</strong>
<span id="demanda">0</span> unidades</p>

                <p><strong>País:</strong> <span
id="pais">-</span></p>

            </div>

        </div>

    </div>

</div>

```

```

</div>

<!-- Componentes del Precio -->

<div class="row mb-4" id="componentesPrecio"
style="display: none;">

    <div class="col-12">

        <div class="card">

            <div class="card-header">

                <h5><i class="fas fa-pie-chart"></i>
Componentes del Precio (Adam Smith)</h5>

            </div>

            <div class="card-body">

                <div class="row">

                    <div class="col-md-4">

                        <div class="componente-precio">

                            <h6><i class="fas
fa-user-tie text-primary"></i> Salarios (Trabajo)</h6>

                            <p class="mb-2">$<span
id="salarios">0</span></p>

                            <div class="progress">

                                <div class="progress-bar
bg-primary" id="salariosBar" style="width: 0%">0%</div>

                            </div>

```

```

        <small
class="text-muted">Remuneración del trabajo</small>

    </div>

</div>

<div class="col-md-4">

    <div class="componente-precio">

        <h6><i class="fas fa-coins
text-success"></i> Beneficios (Capital)</h6>

        <p class="mb-2">$<span
id="beneficios">0</span></p>

        <div class="progress">

            <div class="progress-bar
bg-success" id="beneficiosBar" style="width: 0%">0%</div>

        </div>

        <small
class="text-muted">Remuneración del capital</small>

    </div>

</div>

<div class="col-md-4">

    <div class="componente-precio">

        <h6><i class="fas
fa-mountain text-warning"></i> Rentas (Tierra)</h6>

```

```

                                <p class="mb-2">$<span
id="rentas">0</span></p>

                                <div class="progress">

                                    <div class="progress-bar
bg-warning" id="rentasBar" style="width: 0%">0%</div>

                                </div>

                                <small
class="text-muted">Remuneración de la tierra</small>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

    <!-- Simulador de Mercado -->

    <div class="row mb-4" id="simuladorMercado"
style="display: none;">

        <div class="col-12">

            <div class="card">

                <div class="card-header">

```



```

        <h5><i class="fas fa-sliders-h"></i>
Simulador de Mercado</h5>

    </div>

    <div class="card-body">

        <div class="row">

            <div class="col-md-6">

                <label for="nuevaOferta"
class="form-label">Nueva Oferta:</label>

                <input type="number"
class="form-control" id="nuevaOferta" min="1">

            </div>

            <div class="col-md-6">

                <label for="nuevaDemanda"
class="form-label">Nueva Demanda:</label>

                <input type="number"
class="form-control" id="nuevaDemanda" min="1">

            </div>

        </div>

        <div class="row mt-3">

            <div class="col-12">

                <button class="btn btn-primary"
onclick="simularMercado()">

```

```

                <i class="fas fa-play"></i>
Simular Cambios

            </button>

        </div>

    </div>

    <div id="resultadoSimulacion"
class="mt-3"></div>

    </div>

</div>

</div>

</div>

</div>

    <!-- Historial de Precios -->

    <div class="row mb-4" id="historialPrecios"
style="display: none;">

        <div class="col-12">

            <div class="card">

                <div class="card-header">

                    <h5><i class="fas fa-chart-area"></i>
Historial de Precios</h5>

                </div>

                <div class="card-body">

```

```

        <div class="historial-chart">

            <canvas id="historialChart"
width="400" height="200"></canvas>

        </div>

    </div>

</div>

</div>

</div>

</div>

<!-- Lista de Productos -->

<div class="row">

    <div class="col-12">

        <div class="card">

            <div class="card-header">

                <h5><i class="fas fa-list"></i> Todos
los Productos</h5>

            </div>

            <div class="card-body">

                <div id="listaProductos">

                    <p class="text-muted">Cargando
productos...</p>

                </div>

```

```

        </div>

    </div>

</div>

</div>

<script src="/assets/js/cap6.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootst
rap.bundle.min.js"></script>

    <script src="/assets/js/script.js"></script>
</body>
</html>

```

creando la rutas

Ahora voy a agregar las rutas del Capítulo 6 antes de `router.Run(":8080")`:

```

// ===== ENDPOINTS DEL CAPÍTULO 6 - COMPONENTES DEL PRECIO =====

// Cargar datos del Capítulo 6

handlers.CargarDatosCap6()

// Ruta para la página del simulador del Capítulo 6

router.GET("/cap6", func(c *gin.Context) {

    c.HTML(http.StatusOK, "cap6_simulador.html", gin.H{

```

```
        "title": "Simulador de Componentes del Precio -  
Capítulo 6",  
    })  
})  
  
// Endpoint para obtener todos los productos del Capítulo 6  
router.GET("/api/cap6/productos", handlers.GetProductos)  
  
// Endpoint para obtener un producto específico del Capítulo  
6  
router.GET("/api/cap6/producto/:id", handlers.GetProducto)  
  
// Endpoint para obtener productos por país del Capítulo 6  
router.GET("/api/cap6/productos-pais/:pais",  
handlers.GetProductosPorPais)  
  
// Endpoint para simular cambios en oferta/demanda del  
Capítulo 6  
router.GET("/api/cap6/simular/:id", handlers.SimularMercado)  
  
// Endpoint para obtener indicadores económicos del Capítulo  
6
```

```
router.GET("/api/cap6/indicadores",
handlers.GetIndicadoresEconomicos)

// Endpoint para obtener mercados del Capítulo 6

router.GET("/api/cap6/mercados", handlers.GetMercados)

// Endpoint para análisis de componentes de precio del
Capítulo 6

router.GET("/api/cap6/analizar/:id",
handlers.AnalizarComponentes)
```

creando cap6.css

```
/* Estilos específicos para el Capítulo 6 */

/* Nav blanco */

nav, .navbar-white {

    background-color: white !important;
```

```
color: #333 !important;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

border-bottom: 1px solid #e0e0e0;
}

nav .logo h1 {

    color: #333 !important;
}

nav ul li a {

    color: #333 !important;
}

nav ul li a:hover {

    color: #007bff !important;
}

.hamburger .line {

    background-color: #333 !important;
}
```

```
.menubar {  
  
    background-color: white !important;  
  
    color: #333 !important;  
  
}  
  
.menubar ul li a {  
  
    color: #333 !important;  
  
}  
  
.menubar ul li a:hover {  
  
    color: #007bff !important;  
  
}  
  
/* Estilos adicionales para el simulador */  
  
body {  
  
    background: linear-gradient(135deg, #667eea 0%, #764ba2  
100%);  
  
    min-height: 100vh;  
  
    color: white;  
  
}
```



```
.card {  
    background: rgba(255, 255, 255, 0.1);  
    backdrop-filter: blur(10px);  
    border: 1px solid rgba(255, 255, 255, 0.2);  
    color: white;  
}  
  
.card-header {  
    background: rgba(255, 255, 255, 0.1);  
    border-bottom: 1px solid rgba(255, 255, 255, 0.2);  
}  
  
.table {  
    color: white;  
}  
  
.table th, .table td {  
    border-color: rgba(255, 255, 255, 0.2);  
}  
  
.btn-primary {
```

```
background: linear-gradient(45deg, #667eea, #764ba2);

border: none;
}

.btn-primary:hover {

background: linear-gradient(45deg, #764ba2, #667eea);
}

.componente-precio {

background: rgba(255, 255, 255, 0.1);

border-radius: 10px;

padding: 15px;

margin: 10px 0;
}

.progress {

height: 25px;

background: rgba(255, 255, 255, 0.2);
}

.progress-bar {
```

```
font-weight: bold;
}

.historial-chart {
    background: rgba(255, 255, 255, 0.05);
    border-radius: 10px;
    padding: 20px;
}

/* Responsive */
@media (max-width: 768px) {
    nav {
        padding: 10px;
    }

    nav ul {
        background-color: white !important;
    }

    nav ul li a {
        color: #333 !important;
    }
}
```

```
}  
  
}
```

creando [cap6.js](#)

```
let productos = [];  
  
let productoActual = null;  
  
let historialChart = null;  
  
// Cargar productos al iniciar  
  
document.addEventListener('DOMContentLoaded', function () {  
  
    cargarProductos();  
  
    cargarIndicadoresEconomicos();  
  
});  
  
// Función para cargar productos  
  
async function cargarProductos() {  
  
    try {  
  
        const response = await fetch('/api/cap6/productos');  
  
        const data = await response.json();  
  
  
        if (data.success) {  
  
            productos = data.productos;
```

```

        mostrarListaProductos();

        actualizarSelectProductos();

    }

} catch (error) {

    console.error('Error al cargar productos:', error);

}

}

// Función para mostrar lista de productos
function mostrarListaProductos() {

    const container = document.getElementById('listaProductos');

    let html = '<div class="table-responsive"><table
class="table">';

    html += '<thead><tr><th>Producto</th><th>Precio
Mercado</th><th>Precio
Natural</th><th>Oferta</th><th>Demanda</th><th>País</th><th>Acci
ones</th></tr></thead><tbody>';

    productos.forEach(producto => {

        html += `

            <tr>

                <td><strong>${producto.nombre}</strong></td>

```

```

        <td>${producto.precio_mercado.toFixed(2)}</td>

        <td>${producto.precio_natural.toFixed(2)}</td>

        <td>${producto.oferta}</td>

        <td>${producto.demanda}</td>

        <td>${producto.pais}</td>

        <td>

            <button class="btn btn-sm btn-primary"
onclick="seleccionarProducto('${producto.id}')">

                <i class="fas fa-eye"></i> Ver

            </button>

        </td>

    </tr>

    `;

    });

    html += '</tbody></table></div>';

    container.innerHTML = html;
}

// Función para actualizar select de productos
function actualizarSelectProductos() {

```

```
const select = document.getElementById('productoSelect');

select.innerHTML = '<option value="">Selecciona un
producto...</option>';

productos.forEach(producto => {

    select.innerHTML += `<option
value="${producto.id}">${producto.nombre}
(${producto.pais})</option>`;

});

}

// Función para cargar indicadores económicos
async function cargarIndicadoresEconomicos() {

    try {

        const response = await fetch('/api/cap6/indicadores');

        const data = await response.json();

        if (data.success) {

            mostrarIndicadoresEconomicos(data.indicadores);

        }

    } catch (error) {

        console.error('Error al cargar indicadores:', error);

    }

}
```

```

    }
}

// Función para mostrar indicadores económicos
function mostrarIndicadoresEconomicos(indicadores) {
    const container =
document.getElementById('indicadoresEconomicos');

    let html = '';

    for (const [pais, datos] of Object.entries(indicadores)) {
        html += `
            <div class="mb-3">

                <h6>${pais.charAt(0).toUpperCase() +
pais.slice(1)}</h6>

                <p class="mb-1"><small>Inflación:
${datos.inflacion_anual}%</small></p>

                <p class="mb-1"><small>PIB per cápita:
${datos.pib_per_capita}</small></p>

                <p class="mb-1"><small>Desempleo:
${datos.tasa_desempleo}%</small></p>

                <p class="mb-0"><small>Salario mínimo:
${datos.salario_minimo}</small></p>

            </div>

```



```

        `
    `;

    }

    container.innerHTML = html;
}

// Función para cargar producto específico
async function cargarProducto() {

    const productId =
document.getElementById('productoSelect').value;

    if (!productId) return;

    await seleccionarProducto(productId);
}

// Función para seleccionar producto
async function seleccionarProducto(productId) {

    try {

        const response = await
fetch(`/api/cap6/producto/${productId}`);

        const data = await response.json();
    }
}

```

```
        if (data.success) {

            productoActual = data.producto;

            mostrarProducto(productoActual);

            document.getElementById('productoSelect').value =
productoId;

        }

    } catch (error) {

        console.error('Error al cargar producto:', error);

    }

}

// Función para mostrar producto

function mostrarProducto(producto) {

    // Mostrar secciones

    document.getElementById('infoProducto').style.display =
'block';

    document.getElementById('componentesPrecio').style.display =
'block';

    document.getElementById('simuladorMercado').style.display =
'block';

    document.getElementById('historialPrecios').style.display =
'block';

}
```

```
// Actualizar información básica

document.getElementById('precioNatural').textContent =
producto.precio_natural.toFixed(2);

document.getElementById('precioMercado').textContent =
producto.precio_mercado.toFixed(2);

document.getElementById('diferenciaPrecio').textContent =
(producto.precio_mercado - producto.precio_natural).toFixed(2);

document.getElementById('oferta').textContent =
producto.oferta;

document.getElementById('demanda').textContent =
producto.demanda;

document.getElementById('pais').textContent = producto.pais;

// Actualizar componentes

document.getElementById('salarios').textContent =
producto.componentes.salarios.toFixed(2);

document.getElementById('beneficios').textContent =
producto.componentes.beneficios.toFixed(2);

document.getElementById('rentas').textContent =
producto.componentes.rentas.toFixed(2);

// Calcular porcentajes

const total = producto.precio_mercado;
```

```
const porcentajeSalarios = (producto.componentes.salarios /
total) * 100;

const porcentajeBeneficios =
(producto.componentes.beneficios / total) * 100;

const porcentajeRentas = (producto.componentes.rentas /
total) * 100;

// Actualizar barras de progreso

document.getElementById('salariosBar').style.width =
porcentajeSalarios + '%';

document.getElementById('salariosBar').textContent =
porcentajeSalarios.toFixed(1) + '%';

document.getElementById('beneficiosBar').style.width =
porcentajeBeneficios + '%';

document.getElementById('beneficiosBar').textContent =
porcentajeBeneficios.toFixed(1) + '%';

document.getElementById('rentasBar').style.width =
porcentajeRentas + '%';

document.getElementById('rentasBar').textContent =
porcentajeRentas.toFixed(1) + '%';

// Actualizar valores de simulación
```

```
document.getElementById('nuevaOferta').value =
producto.oferta;

document.getElementById('nuevaDemanda').value =
producto.demanda;

// Crear gráfico de historial

crearGraficoHistorial(producto);
}

// Función para crear gráfico de historial
function crearGraficoHistorial(producto) {

    const ctx =
document.getElementById('historialChart').getContext('2d');

    if (historialChart) {

        historialChart.destroy();

    }

    const labels = [];

    for (let i = 0; i <
producto.historial_precios.natural.length; i++) {

        labels.push(`Período ${i + 1}`);
```

```
}

historialChart = new Chart(ctx, {
  type: 'line',
  data: {
    labels: labels,
    datasets: [{
      label: 'Precio Natural',
      data: producto.historial_precios.natural,
      borderColor: '#4ecdc4',
      backgroundColor: 'rgba(78, 205, 196, 0.1)',
      tension: 0.1
    }, {
      label: 'Precio de Mercado',
      data: producto.historial_precios.mercado,
      borderColor: '#ff6b6b',
      backgroundColor: 'rgba(255, 107, 107, 0.1)',
      tension: 0.1
    }]
  },
  options: {
```

```
responsive: true,

maintainAspectRatio: false,

plugins: {

    title: {

        display: true,

        text: 'Evolución de Precios',

        color: 'white'

    },

    legend: {

        labels: {

            color: 'white'

        }

    }

},

scales: {

    y: {

        ticks: {

            color: 'white'

        },

        grid: {

            color: 'rgba(255, 255, 255, 0.1)'

        }

    }

}
```

```

        }

        },

        x: {

            ticks: {

                color: 'white'

            },

            grid: {

                color: 'rgba(255, 255, 255, 0.1)'

            }

        }

    }

    });
}

// Función para simular mercado
async function simularMercado() {

    if (!productoActual) {

        alert('Selecciona un producto primero');

        return;

    }

```



```
const nuevaOferta =
parseInt(document.getElementById('nuevaOferta').value);

const nuevaDemanda =
parseInt(document.getElementById('nuevaDemanda').value);

if (!nuevaOferta || !nuevaDemanda) {

    alert('Ingresa valores válidos para oferta y demanda');

    return;

}

try {

    const response = await
fetch(`/api/cap6/simular/${productoActual.id}?oferta=${nuevaOferta}&demanda=${nuevaDemanda}`);

    const data = await response.json();

    if (data.success) {

        productoActual = data.producto;

        mostrarProducto(productoActual);

        // Mostrar resultado de simulación
```

```

        const cambios = data.cambios;

document.getElementById('resultadoSimulacion').innerHTML = `
        <div class="alert alert-success">

            <h6>Simulación Completada</h6>

            <p><strong>Precio anterior:</strong>
${cambios.precio_anterior.toFixed(2)}</p>

            <p><strong>Precio nuevo:</strong>
${cambios.precio_nuevo.toFixed(2)}</p>

            <p><strong>Cambio:</strong>
${(cambios.precio_nuevo -
cambios.precio_anterior).toFixed(2)}</p>

        </div>

    `;

    }

    } catch (error) {

        console.error('Error al simular mercado:', error);

        alert('Error al simular mercado');

    }

}

```

resumen

📊 Simulador de Componentes del Precio - Capítulo 6 (Adam Smith)

> **"De los componentes del precio de las mercancías"** - La Riqueza de las Naciones

🎯 Descripción

Simulador interactivo que demuestra los principios económicos del **Capítulo 6** de Adam Smith, mostrando cómo se descompone el precio de las mercancías en sus tres componentes fundamentales:

- **Salarios** (remuneración del trabajo)
- **Beneficios** (remuneración del capital)
- **Rentas** (remuneración de la tierra)

🚀 Características

✨ Funcionalidades Principales

- **Desglose de Precios**: Visualización de los componentes del precio según Adam Smith
- **Simulador de Mercado**: Modificar oferta y demanda para ver cambios en precios
- **Historial de Precios**: Seguimiento de precios natural vs mercado
- **Indicadores Económicos**: Datos por país (Venezuela, Colombia)
- **Análisis Visual**: Gráficos interactivos con Chart.js

📊 Componentes del Precio

``json

```
{  
  "precio_mercado": 250.00,  
  "componentes": {  
    "salarios": 112.50, // 45% - Trabajo  
    "beneficios": 75.00, // 30% - Capital  
    "rentas": 62.50 // 25% - Tierra  
  }  
}
```

...

🛠️ Tecnologías

- **Backend**: Go + Gin Framework
- **Frontend**: HTML5 + CSS3 + JavaScript
- **Gráficos**: Chart.js
- **UI**: Bootstrap 5
- **Iconos**: Font Awesome

📁 Estructura del Proyecto

...

cap6_simulador/

```
|— main.go           # Servidor principal
|— internal/
|   |— handlers/
|   |   |— cap6_handlers.go # API endpoints
|   |   |— database/
|   |       |— data_cap6.json # Datos de productos
|— templates/
|   |— cap6_simulador.html # Interfaz web
|— assets/
|   |— css/
|       |— cap6.css      # Estilos específicos
```

...

🚀 Instalación y Uso

Prerrequisitos

- Go 1.16+
- Navegador web moderno

Ejecutar

```
```bash
```

```
Clonar repositorio
```

```
git clone [url-del-repositorio]
```

```
Navegar al directorio
```

```
cd cap1_division_del_trabajo/cap1_division_del_trabajo
```

```
Ejecutar servidor
```

```
go run main.go
```

```
```
```

Acceder

```
- **URL**: `http://localhost:8081/cap6`
```

```
- **API**: `http://localhost:8081/api/cap6/*`
```

📡 API Endpoints

| Endpoint | Método | Descripción |
|----------|--------|-------------|
|----------|--------|-------------|

| ----- | ----- | ----- |
|-------|-------|-------|
|-------|-------|-------|

| | | |
|-----------------------------------|-----|----------------------------|
| <code>/api/cap6/productos`</code> | GET | Listar todos los productos |
|-----------------------------------|-----|----------------------------|

| | | |
|--------------------------------------|-----|-----------------------------|
| <code>/api/cap6/producto/:id`</code> | GET | Obtener producto específico |
|--------------------------------------|-----|-----------------------------|

| | | |
|-------------------------------------|-----|--------------------------------|
| <code>/api/cap6/simular/:id`</code> | GET | Simular cambios oferta/demanda |
|-------------------------------------|-----|--------------------------------|

| | | |
|-------------------------------------|-----|------------------------|
| <code>/api/cap6/indicadores`</code> | GET | Indicadores económicos |
|-------------------------------------|-----|------------------------|

| \api/cap6/analizar/:id` | GET | Análisis de componentes |

Ejemplo de Uso API

```
```bash
```

```
Obtener productos
```

```
curl http://localhost:8081/api/cap6/productos
```

```
Simular mercado para trigo
```

```
curl "http://localhost:8081/api/cap6/simular/trigo?oferta=1200&demanda=1000"
```

```
```
```

📊 Productos Incluidos

- **Trigo** (Venezuela) - \$250/tonelada
- **Maíz** (Venezuela) - \$180/tonelada
- **Herramientas** (Venezuela) - \$45/unidad
- **Café** (Colombia) - \$320/tonelada
- **Bananas** (Colombia) - \$1.20/kg

🎨 Características de la UI

- **Nav Blanco**: Diseño limpio con navegación blanca
- **Gradiente de Fondo**: Estilo moderno con gradientes
- **Cards Transparentes**: Efecto glassmorphism
- **Responsive**: Adaptable a móviles y tablets
- **Gráficos Interactivos**: Visualización de datos en tiempo real

🛠 Configuración

Variables de Entorno

```
```bash
```

```
Puerto del servidor (opcional)
```

PORT=8080

---

### ### Personalización de Datos

Editar `internal/database/data\_cap6.json` para:

- Agregar nuevos productos
- Modificar precios y componentes
- Cambiar indicadores económicos

## ## 📖 Conceptos Económicos

### ### Precio Natural vs Precio de Mercado

- **Precio Natural**: Costo de producción (salarios + beneficios + rentas)
- **Precio de Mercado**: Determinado por oferta y demanda

### ### Fórmula de Simulación

---

$$\text{Precio Mercado} = \text{Precio Natural} \times (\text{Demanda} / \text{Oferta})$$

---

## ## 📄 Licencia

Este proyecto está bajo la Licencia MIT - ver el archivo [LICENSE](LICENSE) para detalles.

## ## 🧑 Autor


**Adam Smith** - \*Conceptos económicos originales\*

**Desarrollador** - \*Implementación técnica\*

---

muestra cap6

<https://github.com/user-attachments/assets/1681108a-d62f-4482-8c1a-dfc345c6b552>



★ \*\*¡Dale una estrella si te gustó el proyecto!\*\*