

Aquí tienes la documentación actualizada con los cambios de la **Fase 2 (Motor Logístico)** incorporados. He resaltado las modificaciones para que puedas copiar y pegar directamente en tu Google Doc maestro.

1. DOCUMENTO DE DISEÑO DE JUEGO (GDD) - v1.2

Nombre del Proyecto: ElectroNova Inc.

Versión: 1.2 (Post-Implementación de Logística y Tiempos)

Mecánica Central (Core Loop) [ACTUALIZADO]

El ciclo de juego ahora incorpora la dimensión del **TIEMPO**. La producción no es venta inmediata:

1. **Compras:** Adquisición de MP.
2. **Producción:** Transformación de MP en Stock de Fábrica (No vendible aún).
3. **Logística (Envío):** El estudiante decide mover stock de Fábrica a Plaza usando transporte Aéreo o Terrestre.
4. **Tránsito:** La mercancía viaja durante 1 o 2 rondas.
5. **Venta:** Solo se vende lo que ha llegado al almacén de la Plaza.

Sistema Logístico y Tiempos (Lead Times) [NUEVO]

Se introduce la planificación estratégica de envíos. El jugador debe equilibrar costo vs. velocidad:

- **Aéreo (Express):**

- **Tiempo:** 1 Ronda (Llega para la venta de la *siguiente* ronda).
- **Costo:** Alto (\$5.00 / unidad).
- **Uso:** Reacción rápida ante quiebres de stock.

-

- **Terrestre (Estándar):**

- **Tiempo:** 2 Rondas (Llega en la *subsiguiente* ronda).
- **Costo:** Bajo (\$1.00 / unidad).
- **Uso:** Abastecimiento planificado y rentable.

-

Estados del Inventario [NUEVO]

El producto pasa por tres estados físicos:

1. **En Fábrica:** Recién producido. Costo de almacenaje \$0.00 (temporalmente).
2. **En Tránsito:** Viajando. No genera costo de almacén, pero no se puede vender.
3. **En Plaza:** Disponible para el cliente. Genera costo de almacenaje (\$0.20).

2. TECHNICAL DESIGN DOCUMENT (TDD) - v1.2

Esquemas de Datos (Data Modeling) [ACTUALIZADO]

Company Schema (Empresa):

codeJavaScript

```
{
  financials: { cash: Decimal128, ... },
  rawMaterials: { units: Number },
  // [NUEVO] Stock retenido en origen
  factoryStock: {
    units: Number,
    unitCost: Decimal128
  },
  // [NUEVO] Array de envíos viajando
  inTransit: [
    {
      batchId: String,
      units: Number,
      method: String, // 'Aereo' | 'Terrestre'
      roundsRemaining: Number, // Contador regresivo (2 -> 1 -> 0)
      unitCost: Decimal128
    }
  ],
  inventory: [ /* Stock en Plaza (Vendible) */ ]
}
```

Decision Schema (Decisión):

codeJavaScript

```

{
  // ... (Precio, Marketing, Producción, Compras)
  // [ACTUALIZADO] Estructura Logística Activa
  logistics: [
    {
      destination: String, // 'Plaza Central'
      units: Number,
      method: String // 'Aereo' | 'Terrestre'
    }
  ]
}

```

Algoritmo de Procesamiento de Ronda (Logic Flow) [ACTUALIZADO]

El servidor ejecuta la ronda siguiendo una secuencia cronológica estricta para respetar los tiempos de llegada:

1. Fase de Llegadas (Inbound):

- Se revisa `inTransit`. Si `roundsRemaining <= 1` la carga se mueve a `inventory` (Plaza).
- Si `roundsRemaining > 1`, se resta 1 al contador.

2.

3. Fase Operativa (Factory):

- Compra de MP (Caja
- ↓
- ↓
- , MP
- ↑
- ↑
-).
- Producción: Se consume MP y se genera `factoryStock` [NOTA:] Lo producido se queda en fábrica, no va a venta.

4.

5. Fase Logística (Outbound):

- Se procesan las órdenes de envío `decision.logistics`.
- Se valida stock en fábrica.
- Se mueve de `factoryStock`

- →
- →
- `inTransit`.
- Se cobra el flete (Caja
- ↓
- ↓
-).

6.

7. Fase de Mercado (Sales):

- El motor ECPCIM calcula ventas basándose **solo** en el stock disponible en `inventory`(Plaza).

8.

3. BIBLIA DE ESTRATEGIA EDTECH (Guía del Estudiante)

Guía de Operaciones [ACTUALIZADO]

- **La Regla del Tiempo (Lead Time):** En ElectroNova, nada es instantáneo. Lo que produces hoy, no lo puedes vender hoy. Debes enviarlo a la plaza comercial.
- **Planificación Logística:**
 - Si envías por **Avión (\$5/u)**, la mercancía estará disponible al inicio de la **siguiente ronda**.
 - Si envías por **Camión (\$1/u)**, la mercancía tardará **dos rondas** en llegar.
 - *Estrategia:* Usa camión para el volumen grueso (barato) y avión solo para emergencias. Si te quedas sin stock en la Plaza, perderás ventas aunque tengas miles de unidades en la Fábrica.

●

Estructura de Costos Completa [ACTUALIZADO]

- **Materia Prima:** \$15.00 / u.
- **Manufactura:** \$35.00 / u.
- **Logística Aérea:** \$5.00 / u.
- **Logística Terrestre:** \$1.00 / u.

- **Almacenaje (Plaza):** \$0.20 / u / ronda.

4. PLAN DE DESARROLLO ACTUALIZADO (ROADMAP v1.3)

Estado Actual: Versión 1.2 (Logística Implementada).

Meta: Segmentación Geográfica.

FASE 1: Cadena de Suministro Básica (COMPLETADA ✓)

- ✓ Implementación de Materia Prima.
- ✓ Restricción de producción.
- ✓ Barrera de precios.

FASE 2: Logística y Tiempos (COMPLETADA ✓)

- ✓ Modelo de datos para `inTransit` y `factoryStock`.
- ✓ Algoritmo de procesamiento de llegadas y envíos.
- ✓ UI de decisiones con selector de transporte.
- ✓ Dashboard con visualización de cadena completa (Fábrica -> Tránsito -> Plaza).

FASE 3: Segmentación de Mercado (PRÓXIMO OBJETIVO)

Objetivo: Romper el mercado único "Plaza Central" en 3 zonas.

- **Tarea 3.1 (Data):** Configurar `GlobalConfig` con 3 plazas:
 - *Norte:* Alta demanda, alta calidad, paga bien.
 - *Sur:* Demanda media, sensible al precio.
 - *Centro:* Mercado balanceado.
- **Tarea 3.2 (Logística):** Actualizar `DecisionModal` para permitir envíos a múltiples destinos (ej. "Enviar 500 al Norte por Avión, 500 al Sur por Tierra").
- **Tarea 3.3 (Motor):** Refactorizar `marketEngine` para calcular cuotas de mercado independientes por cada plaza.
- **Tarea 3.4 (Frontend):** Desglosar el Dashboard para ver inventarios y ventas por región.

FASE 4: Pulido y Eventos (Pendiente)

- **Tarea 4.1:** Eventos aleatorios (Huelgas logísticas).
- **Tarea 4.2:** Reportes financieros avanzados.