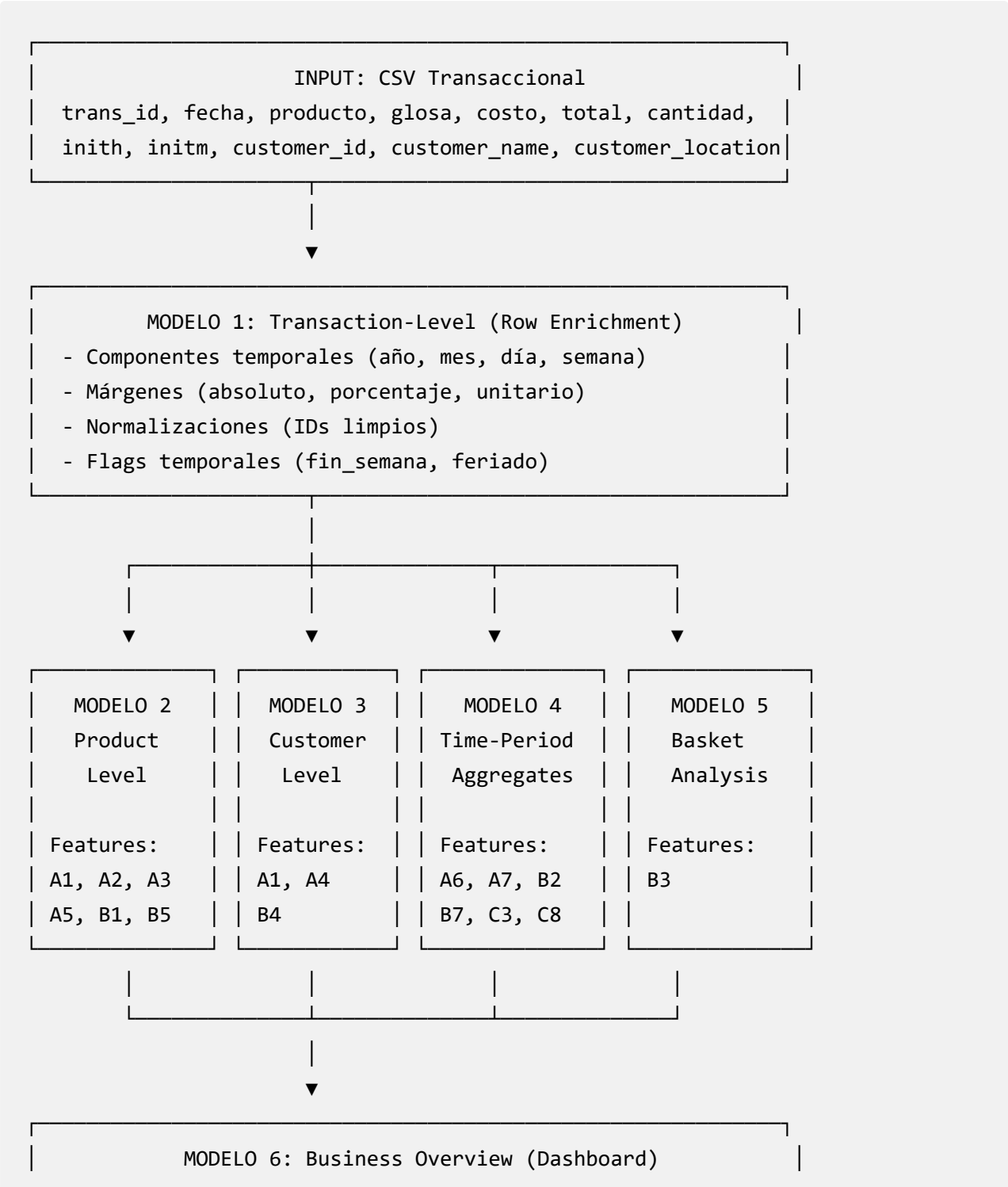


# GabeDA - Arquitectura Completa de Modelos de Datos

## Resumen Ejecutivo

Este documento describe la arquitectura completa de modelos analíticos de GabeDA, organizada en **6 modelos** con diferentes niveles de agregación. Cada modelo soporta múltiples features del menú de capacidades analíticas.

## Arquitectura General



- Consolida insights de todos los modelos
- KPIs principales y health score
- Alertas críticas priorizadas
- Oportunidades identificadas
Features: B6, Base para C2, C4, C6

## Mapeo de Features por Modelo

### Modelo 1: Transaction-Level

**Tipo:** Row enrichment (FILTERS stage)

**Aggregation:** Ninguna - opera fila por fila

**Features:** Base para todos los demás modelos

### Modelo 2: Product-Level

**Aggregation:** GROUP BY producto\_clean

**Features:**

- ✓ **A1** - Análisis Pareto (productos)
- ✓ **A2** - Alertas de Inventario
- ✓ **A3** - Salud de Inventario
- ✓ **A5** - Velocidad de Productos
- ✓ **B1** - Rentabilidad por Producto
- ✓ **B5** - Punto de Reorden Inteligente

### Modelo 3: Customer-Level

**Aggregation:** GROUP BY customer\_id\_clean

**Features:**

- ✓ **A1** - Análisis Pareto (clientes)
- ✓ **A4** - Segmentación de Clientes
- ✓ **B4** - Detección de Clientes en Riesgo

### Modelo 4: Time-Period

**Aggregation:** GROUP BY fecha/mes/año/trimestre

**Features:**

- ✓ **A6** - Pronóstico de Ventas Básico
- ✓ **A7** - Análisis de Tendencias Temporales
- ✓ **B2** - Análisis de Estacionalidad
- ✓ **B7** - Comparación con Períodos Anteriores
- **C3** - Alertas Predictivas (base temporal)
- **C8** - Pronósticos Multi-Factor (base temporal)

## Modelo 5: Product-Basket

**Aggregation:** GROUP BY basket\_id y pares de productos





**Features:**

-  **B3** - Análisis de Canastas (Market Basket)

## Modelo 6: Business Overview

**Aggregation:** Business-wide (consolidación)

**Features:**

-  **B6** - Dashboard de Indicadores Clave
  -  Base para **C2** - Explicaciones Inteligentes
  -  Base para **C4** - Recomendaciones Automáticas
  -  Base para **C6** - Coach de Negocios
- 

## Features con IA (Capas Adicionales)

Features de IA que NO son modelos de datos:

### C1 - Asistente de Voz/Chat

- **Tipo:** Interface layer
- **Consume:** Todos los modelos (2-6)
- **Tecnología:** LLM + NLP

### C2 - Explicaciones Inteligentes

- **Tipo:** Interpretation layer sobre Modelo 6
- **Consume:** Business Overview + contexto
- **Tecnología:** LLM

### C3 - Alertas Predictivas

- **Tipo:** Enhancement del Modelo 4
- **Base:** Time-Period metrics
- **Tecnología:** ML forecasting (ARIMA, Prophet, LSTM)

### C4 - Recomendaciones Automáticas

- **Tipo:** Action layer sobre Modelo 6
- **Consume:** Business Overview + reglas de negocio
- **Tecnología:** LLM + Rule engine

### C5 - Procesamiento de Documentos (OCR)

- **Tipo:** Input enhancement
- **Posición:** Antes del Modelo 1
- **Tecnología:** OCR + NLP

## C6 - Coach de Negocios Personalizado

- **Tipo:** Conversational layer
- **Consume:** Todos los modelos + contexto conversacional
- **Tecnología:** LLM con RAG

## C7 - Detección de Anomalías

- **Tipo:** Enhancement de Modelos 2, 3, 4
- **Base:** Time-series de métricas
- **Tecnología:** Isolation Forest, Autoencoders

## C8 - Pronósticos Multi-Factor

- **Tipo:** Enhancement del Modelo 4
- **Base:** Time-Period + features engineered
- **Tecnología:** ML (XGBoost, LightGBM) + ensemble

---

## Flujo de Ejecución Completo

```
# Pipeline completo de GabeDA
def run_gabeda_pipeline(csv_file, stock_file=None, config={}):
    """
    Ejecuta pipeline completo de análisis GabeDA

    Parameters:
    -----
    csv_file : str
        Path al CSV transaccional
    stock_file : str, optional
        Path al CSV con stock actual
    config : dict
        Configuraciones (lead_time, umbrales, etc.)
    """

    # PASO 1: Cargar datos
    print("=" * 80)
    print("GABEDA ANALYTICS PIPELINE")
    print("=" * 80)

    df_raw = pd.read_csv(csv_file)
    print(f"✓ Cargados {len(df_raw):,} registros")

    if stock_file:
        stock_df = pd.read_csv(stock_file)
        print(f"✓ Cargado inventario: {len(stock_df)} productos")
    else:
```

```

stock_df = None
print("⚠ Sin datos de inventario actual")

# PASO 2: Modelo 1 - Transaction Enrichment
print("\n[1/6] Ejecutando Modelo 1: Transaction Enrichment...")
df_enriched = enrich_transactions(df_raw)
print(f" ✓ {len(df_enriched.columns)} columnas (añadidas {len(df_enriched.columns) - len(df_raw.columns)} columnas)")

# PASO 3: Modelo 2 - Product Level
print("\n[2/6] Ejecutando Modelo 2: Product-Level Analysis...")
product_model = calculate_product_model(
    df_enriched,
    stock_df=stock_df,
    lead_time=config.get('lead_time', 10)
)
print(f" ✓ Analizados {len(product_model)} productos")

# PASO 4: Modelo 3 - Customer Level
print("\n[3/6] Ejecutando Modelo 3: Customer-Level Analysis...")
customer_model = calculate_customer_model(df_enriched)
print(f" ✓ Analizados {len(customer_model)} clientes")
print(f" ✓ Segmentos: {customer_model['segmento'].value_counts().to_dict()}")

# PASO 5: Modelo 4 - Time Period
print("\n[4/6] Ejecutando Modelo 4: Time-Period Analysis...")
time_models = calculate_time_period_models(df_enriched)
print(f" ✓ Métricas diarias: {len(time_models['daily'])} días")
print(f" ✓ Métricas mensuales: {len(time_models['monthly'])} meses")

# PASO 6: Modelo 5 - Market Basket (opcional)
print("\n[5/6] Ejecutando Modelo 5: Market Basket Analysis...")
try:
    market_basket = calculate_market_basket_model(
        df_enriched,
        time_window_hours=config.get('basket_window', 4),
        min_support=config.get('min_support', 0.01),
        min_confidence=config.get('min_confidence', 0.3),
        min_lift=config.get('min_lift', 1.2)
    )
    print(f" ✓ {len(market_basket['strong_rules'])} reglas de asociación fueron encontradas")
    print(f" ✓ {len(market_basket['natural_bundles'])} bundles naturales identificados")
except Exception as e:
    print(f" ⚠ Market Basket no disponible: {str(e)}")
    market_basket = None

# PASO 7: Modelo 6 - Business Overview
print("\n[6/6] Ejecutando Modelo 6: Business Overview...")

```

```

business_overview = calculate_business_overview_model(
    df_enriched,
    product_model,
    customer_model,
    time_models['monthly'],
    market_basket
)

dashboard = business_overview['executive_dashboard']
print(f"  ✓ Health Score: {dashboard['salud_general']['score']}/100 - {dashb
print(f"  ✓ Alertas Críticas: {dashboard['salud_general']['alertas_criticas']
print(f"  ✓ Oportunidades: {dashboard['salud_general']['oportunidades_identi

# RESUMEN FINAL
print("\n" + "=" * 80)
print("PIPELINE COMPLETADO".center(80))
print("=" * 80)
print_executive_summary(dashboard)

# Retornar todos los modelos
return {
    'df_enriched': df_enriched,
    'product_model': product_model,
    'customer_model': customer_model,
    'time_models': time_models,
    'market_basket': market_basket,
    'business_overview': business_overview
}

# EJECUCIÓN
results = run_gabeda_pipeline(
    csv_file='comercializadora_transactions.csv',
    stock_file='inventario_actual.csv', # opcional
    config={
        'lead_time': 10,
        'basket_window': 4,
        'min_support': 0.01,
        'min_confidence': 0.3,
        'min_lift': 1.2
    }
)

```

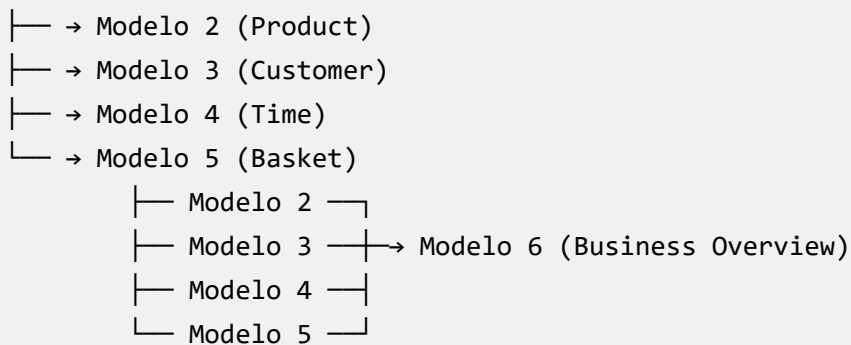
## Frecuencia de Actualización Recomendada

Modelo	Frecuencia	Razón
--------	------------	-------

Modelo 1	On-demand	Se ejecuta cuando hay nuevos datos
Modelo 2	Mensual	Métricas de producto cambian lentamente
Modelo 3	Mensual	Segmentación estable, excepto churn
Modelo 3 (Churn)	Semanal	Solo sección de riesgo para alertas
Modelo 4 (Daily)	Diario	Monitoreo operacional
Modelo 4 (Monthly)	Mensual	Reportes ejecutivos
Modelo 5	Mensual	Asociaciones cambian lentamente
Modelo 6	Semanal	Dashboard ejecutivo

## Dependencias entre Modelos

Modelo 1 (Transaction Enriched)



### Implicación:

- Modelo 1 debe ejecutarse primero siempre
- Modelos 2-5 pueden ejecutarse en paralelo
- Modelo 6 requiere que 2-5 estén completos

## Almacenamiento de Outputs

Estructura de Archivos Recomendada:

```

gabeda_data/
├── raw/
│   └── comercializadora_transactions.csv
├── processed/
│   └── transactions_enriched.parquet
├── models/
│   ├── product_model.parquet
│   ├── customer_model.parquet
│   ├── daily_metrics.parquet
│   └── monthly_metrics.parquet

```

```
|   |— weekly_metrics.parquet
|   |— quarterly_metrics.parquet
|   |— seasonality_indices.json
|   |— market_basket_rules.parquet
|   |— business_overview.json
|— reports/
|   |— dashboard_2025_10_08.html
```

## Formatos Recomendados:

- **Parquet:** Para DataFrames grandes (eficiente, comprimido)
- **JSON:** Para dashboards y configuraciones
- **CSV:** Para exports a Excel/otras herramientas
- **HTML/PDF:** Para reportes ejecutivos

---

## Configuración Global

```
# config.yaml
pipeline:
  lead_time_days: 10
  z_score_service_level: 1.65 # 95%

thresholds:
  stock_alert_days: 7
  no_movement_days: 90
  churn_risk_deviation: 1.5
  min_margin_percent: 15

market_basket:
  basket_window_hours: 4
  min_support: 0.01
  min_confidence: 0.3
  min_lift: 1.2

dashboard:
  health_score_weights:
    growth: 0.35
    margin: 0.25
    trend: 0.25
    retention: 0.15
```

---

## Testing y Validación



```
def validate_pipeline(results):
    """Valida integridad del pipeline completo"""

    checks = {
        'transaction_enrichment': len(results['df_enriched']) > 0,
        'products_analyzed': len(results['product_model']) > 0,
        'customers_analyzed': len(results['customer_model']) > 0,
        'time_series': len(results['time_models']['monthly']) >= 2,
        'dashboard_generated': 'executive_dashboard' in results['business_overview']
    }

    # Validaciones de consistencia
    total_ingresos_enriched = results['df_enriched']['total'].sum()
    total_ingresos_product = results['product_model']['total_ingresos'].sum()

    assert abs(total_ingresos_enriched - total_ingresos_product) < 1, \
        "Inconsistencia en totales entre modelos"

    print("✓ Todas las validaciones pasaron")
    return all(checks.values())
```

## Roadmap de Implementación

### Fase 1: Foundation (Semanas 1-2)

- ✓ Modelo 1: Transaction Enrichment
- ✓ Modelo 2: Product-Level (sin inventario avanzado)
- ✓ Modelo 3: Customer-Level (RFM básico)
- ✓ Modelo 4: Monthly Time-Period

### Fase 2: Advanced Analytics (Semanas 3-4)


- ✓ Modelo 2: Features avanzados (rotación, reorden)
- ✓ Modelo 3: Churn detection
- ✓ Modelo 4: Daily/Weekly/Seasonality
- ✓ Modelo 5: Market Basket

### Fase 3: Business Intelligence (Semana 5)




- ✓ Modelo 6: Business Overview
- ✓ Alertas críticas
- ✓ Identificación de oportunidades

### Fase 4: AI Layer (Semanas 6-8)



- C2: Explicaciones con LLM
- C5: OCR para input

-  C3: Alertas predictivas con ML

#### Fase 5: Advanced AI (Semanas 9-12)

-  C8: Pronósticos avanzados
-  C7: Detección de anomalías
-  C4: Recomendaciones automáticas

#### Fase 6: AI Interface (Semanas 13-16)

-  C1: Chat interface
-  C6: Business coach

---

## Métricas de Performance

Modelo	Registros	Tiempo Esperado	Memoria
Modelo 1	50k	2-5 seg	100 MB
Modelo 2	50k → 500 productos	5-10 seg	50 MB
Modelo 3	50k → 1000 clientes	10-15 seg	80 MB
Modelo 4	50k → 24 meses	5-10 seg	30 MB
Modelo 5	50k → 10k pares	30-60 seg	200 MB
Modelo 6	Consolidación	2-5 seg	20 MB
<b>Total</b>	<b>50k registros</b>	<b>~2 min</b>	<b>~500 MB</b>

---

## Próximos Pasos

1. **Implementar Modelo 1** completo con todas las transformaciones
  2. **Testing** con CSV real de comercializadora
  3. **Optimización** de queries y agregaciones
  4. **Documentación** de campos calculados para cada modelo
  5. **API/Interface** para ejecutar pipeline on-demand
  6. **Visualizaciones** para cada modelo
  7. **Alertas automatizadas** vía email/WhatsApp
- 

## Contacto y Soporte

Para preguntas sobre la arquitectura:

- **Documentación:** Ver documentos individuales de cada modelo
- **Issues:** Reportar en repositorio del proyecto
- **Updates:** Documentación vive - actualizar según evolucione el proyecto