# Especificación Técnica y Estratégica para la Implementación de Inteligencia de Negocios en Operaciones de Seguridad: Ciudad Juan Bosch

## Resumen Ejecutivo

La gestión de la seguridad en entornos urbanos planificados de alta densidad, como el proyecto Ciudad Juan Bosch (CJB), requiere una transición fundamental desde la recopilación pasiva de datos hacia la inteligencia táctica activa. Actualmente, la infraestructura de información de la Dirección de Seguridad de CJB se basa en la captura de incidentes mediante Microsoft Forms, un mecanismo eficiente para el registro digital pero insuficiente para el análisis estratégico en tiempo real. La acumulación de registros en archivos de Excel, con columnas ricas en detalles operativos como "Cuadrante", "Tipo de Incidente" y "Hora de inicio" 1, representa un activo de inteligencia latente que, sin las herramientas de visualización adecuadas, permanece subutilizado.

El presente informe detalla la arquitectura, el diseño y la estrategia de implementación de un **Tablero de Control Dinámico (Dashboard)** basado en tecnologías web del lado del cliente (HTML5, JavaScript, dc.js y Crossfilter). Este sistema está diseñado para replicar la experiencia interactiva de herramientas empresariales como Power BI, permitiendo a la Dirección Social y a los mandos operativos filtrar, segmentar y analizar datos instantáneamente mediante la carga directa de archivos Excel, sin depender de infraestructura de servidores complejos ni licencias recurrentes. A través de la visualización de datos interconectada, el sistema transformará filas estáticas de incidentes en mapas de calor operativos, curvas de tendencia temporal y perfiles de desempeño de oficiales, permitiendo una asignación de recursos basada en evidencia empírica.

## 1. Contexto Operacional y Taxonomía de Datos en Ciudad Juan Bosch

Para diseñar una herramienta de inteligencia efectiva, es imperativo comprender primero el ecosistema operativo de Ciudad Juan Bosch y la naturaleza de los datos que genera su aparato de seguridad. CJB no es simplemente un conjunto residencial, sino una "ciudad satélite" con una zonificación específica que influye directamente en la tipología del crimen y las infracciones administrativas.

### 1.1 El Paisaje de Seguridad y la Zonificación por Cuadrantes

La estructura de seguridad de CJB opera sobre una lógica territorial definida por "Cuadrantes" (e.g., B1, B2, B3, B4). Estos cuadrantes no son meras etiquetas geográficas; representan microclimas de seguridad con características distintas. Por ejemplo, los datos preliminares sugieren que ciertos cuadrantes como el B1 pueden concentrar una mayor densidad de incidentes relacionados con el tránsito y la seguridad municipal, mientras que áreas periféricas en el B2 o B4 podrían presentar una mayor incidencia de controles migratorios y detenciones de indocumentados.1

El análisis de los registros cargados revela que la seguridad en CJB es un esfuerzo multidimensional que involucra a la Policía Nacional, la DIGESETT (tránsito) y unidades especializadas de control migratorio. La presencia de campos como "Oficial a cargo1" y "Rol de la Persona" indica que cada incidente es un evento complejo con múltiples actores. Un tablero estático que solo cuente "incidentes" fallaría en capturar la carga real de trabajo; por ende, el sistema propuesto debe ser capaz de desglosar estos eventos, contabilizando no solo el hecho, sino la cantidad de recursos humanos desplegados y la cantidad de civiles involucrados.

### 1.2 Anatomía del Conjunto de Datos y Desafíos de Ingesta

La viabilidad técnica del dashboard depende de la correcta interpretación del esquema de datos exportado por Microsoft Forms. El archivo "Registro Rápido de Incidentes (SEGURIDAD).csv" presenta una estructura tabular rica pero que requiere una normalización rigurosa antes de ser visualizada.

La Dimensión Temporal:

Las columnas Hora de inicio y Hora de finalización son los pilares de cualquier análisis de tendencias. El formato observado en los datos es D/M/YYYY HH:MM (ejemplo: "25/8/2025 15:22").1 Este formato presenta un desafío técnico específico: los navegadores web y las librerías estándar de JavaScript suelen esperar formatos ISO (YYYY-MM-DD) o formatos americanos (MM/DD/YYYY). Si el dashboard no implementa un analizador sintáctico (parser) personalizado utilizando la librería D3.js con el especificador exacto %d/%m/%Y %H:%M 2, los gráficos de tiempo fallarán, impidiendo que la Dirección Social identifique las "horas pico" de criminalidad o accidentalidad.

Jerarquía de Tipificación de Incidentes:

La columna Tipo de Incidente actúa como la categoría primaria, conteniendo valores como "Migración", "DIGESETT" o "Seguridad Municipal". Sin embargo, existen columnas adyacentes que funcionan como subcategorías o modificadores, tales como Incidentes relacionados a tránsito (que puede contener "Accidente") o Cantidad de Indocumentados detenidos. La lógica del dashboard debe ser capaz de fusionar o jerarquizar estas columnas. Por ejemplo, un registro marcado como "DIGESETT" en el tipo principal y "Accidente" en la columna de tránsito debe ser visualizado prioritariamente como un Accidente para diferenciarlo de una fiscalización de rutina.1

Complejidad de Múltiples Actores:

Uno de los aspectos más complejos del dataset es la estructura de "múltiples involucrados". El formulario permite registrar hasta cuatro personas adicionales (Rol de la Persona1 a Rol de la Persona4), cada una con su nombre y cédula. En un análisis plano de Excel, estas son columnas separadas. Para un dashboard de inteligencia, esto requiere una transformación de datos llamada "unpivoting" o normalización en memoria. El sistema debe ser capaz de iterar sobre estas columnas para calcular métricas como "Total de Personas Detenidas" o "Total de Víctimas Asistidas", independientemente de si fueron registradas como la persona principal o la cuarta persona involucrada.1

## 2. Arquitectura Técnica: La Experiencia "Power BI" en el Navegador

La solicitud de la Dirección Social es clara: se requiere una funcionalidad similar a Power BI (filtrado cruzado, interactividad) pero ejecutada desde un archivo HTML local cargando un Excel. Esto descarta las arquitecturas tradicionales basadas en servidor (como SQL Server + Tableau) y exige una arquitectura "Client-Side" (del lado del cliente) robusta.

### 2.1 El Núcleo de Procesamiento: Crossfilter y dc.js

Para lograr la interactividad fluida donde al hacer clic en un gráfico de "Tipos de Incidente" se actualice instantáneamente el mapa de "Cuadrantes", la solución técnica óptima es la combinación de las librerías **Crossfilter.js** y **dc.js**.

* **Crossfilter como Motor de Indexación:** A diferencia de las bases de datos tradicionales que requieren consultas SQL lentas para cada filtro, Crossfilter carga todo el conjunto de datos del Excel en la memoria RAM del navegador y crea índices multidimensionales. Esto permite filtrar conjuntos de datos de hasta cientos de miles de registros en menos de 10 milisegundos.4 Cuando el usuario selecciona el cuadrante "B1", Crossfilter no vuelve a consultar una base de datos; simplemente ajusta los punteros de memoria para reflejar solo los registros de B1 en todas las demás dimensiones (tiempo, tipo, oficial).
* **dc.js como Capa de Visualización:** dc.js es una librería de gráficos diseñada específicamente para trabajar sobre Crossfilter. Utiliza **D3.js** (Data-Driven Documents) para renderizar gráficos vectoriales (SVG) que son nítidos en cualquier resolución.6 La ventaja crítica de dc.js sobre otras librerías como Chart.js es que los gráficos de dc.js están "vivos" y conectados entre sí por defecto; no se requiere escribir código complejo para vincular el gráfico de barras con el gráfico de pastel.

### 2.2 Estrategia de Ingesta de Datos: SheetJS (xlsx)

Dado que la fuente de datos es un archivo local (Excel/CSV exportado de Microsoft Forms), el navegador debe tener la capacidad de leer sistemas de archivos binarios. La librería **SheetJS (xlsx)** es el estándar industrial para esta tarea.

El flujo de trabajo técnico diseñado es el siguiente:

1. El usuario arrastra el archivo "Registro Rápido de Incidentes.xlsx" al área de carga del dashboard HTML.
2. Un FileReader de HTML5 lee el archivo como un ArrayBuffer binario.
3. SheetJS analiza este buffer y convierte la hoja de cálculo activa en un objeto JSON (JavaScript Object Notation).7
4. Este objeto JSON es sometido a una limpieza automática (parsing de fechas, eliminación de filas vacías) antes de ser inyectado en el motor Crossfilter.

Este enfoque tiene una ventaja estratégica de seguridad de la información: **Privacidad Total**. Al no haber un servidor backend, los datos sensibles sobre detenidos y operaciones policiales nunca salen de la computadora del analista. Todo el procesamiento ocurre localmente en su navegador, eliminando riesgos de interceptación de datos en la nube.

### 2.3 Diseño Responsivo e Interfaz de Usuario

Para asegurar que el dashboard sea utilizable tanto en los monitores del centro de comando como en las tabletas de los supervisores de campo, se utilizará el framework **Bootstrap 5**. Bootstrap proporciona un sistema de rejilla (grid) fluido que reorganizará los gráficos automáticamente según el tamaño de la pantalla.9 Además, se implementará un esquema de colores de "Alto Contraste" o "Modo Oscuro", preferido en entornos de monitoreo de seguridad (SOC) para reducir la fatiga visual durante turnos prolongados.

## 3. Estrategia de Visualización y Métricas Clave

El diseño del dashboard no debe ser una mera colección de gráficos, sino una herramienta de respuesta a preguntas estratégicas. Basado en las columnas disponibles en el Microsoft Form, se proponen los siguientes componentes visuales interconectados.

### 3.1 Cronología de Operaciones (Análisis Temporal)

Pregunta Estratégica: ¿En qué momentos del día o de la semana es más vulnerable la ciudad?

Implementación: Un gráfico de barras de serie temporal (dc.barChart) vinculado a la dimensión Hora de inicio.

* **Funcionalidad:** Este gráfico permitirá hacer "brushing" (arrastrar el ratón para seleccionar un rango). Al seleccionar, por ejemplo, el horario de 18:00 a 22:00, el resto del dashboard se filtrará para mostrar qué tipos de incidentes y en qué cuadrantes ocurren específicamente en esas horas nocturnas. Esto es vital para ajustar los turnos de patrullaje de la Policía Nacional y la vigilancia privada.11

### 3.2 Mapa de Calor Operativo (Distribución por Cuadrantes)

Pregunta Estratégica: ¿Dónde debemos concentrar la fuerza de tarea?

Implementación: Un gráfico de filas (dc.rowChart) o un mapa de calor (dc.heatMap) basado en la columna Cuadrante donde sucedió el hecho.

* **Análisis:** Los datos preliminares muestran una segmentación clara (B1, B2, etc.). Un gráfico de filas ordenado descendentemente mostrará inmediatamente qué cuadrante es el más conflictivo. La capacidad de filtrado cruzado permitirá, al hacer clic en "B2", ver instantáneamente qué oficiales están asignados allí y qué tipos de incidentes predominan (e.g., si B2 sufre más de robos o de accidentes de tránsito).13

### 3.3 Tipología de Incidentes (Distribución Categórica)

Pregunta Estratégica: ¿Cuál es la naturaleza de la amenaza predominante?

Implementación: Un gráfico de anillo (dc.pieChart con radio interno) basado en Tipo de Incidente.

* **Lógica de Negocio:** Es crucial distinguir entre "Seguridad Policial" (crimen), "DIGESETT" (tránsito) y "Migración". El dashboard debe utilizar colores semánticos: Rojo para seguridad policial/crimen, Amarillo para tránsito, Azul para migración. Esto permite una lectura cognitiva rápida del estado de la ciudad. Además, se debe implementar una lógica de "drill-down" para que, si se selecciona "Tránsito", se pueda ver el sub-desglose de "Incidentes relacionados a tránsito" (choques, fiscalización, etc.).14

### 3.4 Desempeño y Carga de Oficiales

Pregunta Estratégica: ¿Está equitativamente distribuida la carga de trabajo entre los oficiales?

Implementación: Un gráfico de filas (dc.rowChart) basado en la columna Oficial a cargo1.

* **Insight:** Este componente revelará quiénes son los oficiales que reportan más incidentes. Un desequilibrio aquí podría indicar dos cosas: un oficial altamente proactivo que merece reconocimiento, o un sector asignado con una carga de trabajo insostenible que requiere refuerzos. Al cruzar esto con el "Tipo de Incidente", la Dirección puede ver si el Tte. Gómez Rosado, por ejemplo, está manejando desproporcionadamente casos de migración frente a otros tipos.1

### 3.5 Registro Detallado Interactivo

Pregunta Estratégica: ¿Cuáles son los detalles específicos de los eventos filtrados?

Implementación: Una tabla de datos (dc.dataTable) al pie del dashboard.

* **Innovación:** A diferencia de una tabla de Excel estática, esta tabla solo mostrará las filas que coincidan con los filtros activos en los gráficos. Incluirá las columnas críticas: Hora, Cuadrante, Narrativa del Incidente y Acciones Tomadas.
* **Manejo de Evidencia:** La columna Evidencia Visual contiene URLs a SharePoint. El dashboard debe renderizar esto no como texto, sino como hipervínculos activos <a>Ver Evidencia</a> o incluso previsualizaciones de imagen en miniatura si los permisos de SharePoint lo permiten, facilitando la auditoría visual inmediata.15

## 4. Normalización y Limpieza de Datos Automatizada

El éxito del dashboard radica en su capacidad para manejar datos "sucios" o inconsistentes provenientes de la entrada manual en Microsoft Forms. El script de carga debe incluir una capa de saneamiento robusta.

### 4.1 Tratamiento de Múltiples Involucrados

Como se identificó en el análisis de los datos 1, la estructura horizontal de "Persona1, Persona2, Persona3" es ineficiente para el conteo.

* **Solución Algorítmica:** Durante la ingesta de datos con SheetJS, el script ejecutará un bucle que recorrerá las columnas de personas. Si Rol de la PersonaN no está vacío, incrementará un contador interno Total\_Involucrados para ese incidente. Esto permitirá crear una métrica (KPI) de "Total de Personas Procesadas", que es un indicador de impacto más fuerte que el simple conteo de incidentes.

### 4.2 Estandarización de Texto

Los campos de texto libre como Cuadrante pueden contener errores humanos (e.g., "b1", "B1 ", "Cuadrante B1").

* **Lógica de Limpieza:** El sistema normalizará automáticamente estas entradas a mayúsculas y eliminará espacios en blanco (trimming) para asegurar que "b1" y "B1" se agrupen correctamente en el mismo segmento del gráfico.

### 4.3 Parsing de Acciones Tomadas

La columna Acciones Tomadas a menudo contiene múltiples valores separados por punto y coma (e.g., "Se solicitó refuerzo; Se realizó arresto").

* **Desglose Analítico:** Para visualizar esto correctamente, se utilizará una función de reducción personalizada en Crossfilter. Esta función separará las cadenas de texto por el delimitador ";" y contará cada acción por separado. Esto permitirá responder preguntas como "¿Cuántas veces se solicitó refuerzo en total?", incluso si esa acción fue parte de un registro con múltiples otras acciones.17

## 5. Guía de Implementación Técnica Paso a Paso

A continuación, se presenta la especificación técnica para que el equipo de desarrollo o el analista principal proceda con la construcción.

### 5.1 Estructura del Archivo HTML

El entregable será un único archivo HTML (dashboard\_cjb.html) que contiene la estructura, los estilos (CSS) y la lógica (JS). Esto facilita su distribución por correo o SharePoint sin necesidad de instalación.

Dependencias (CDNs):

Se deben incluir las siguientes librerías en el <head> del documento:

1. **Bootstrap 5 (CSS/JS):** Para el diseño responsivo y estética moderna.19
2. **D3.js (Versión 7):** Motor gráfico base.20
3. **Crossfilter2:** Motor de filtrado multidimensional.21
4. **dc.js (Versión 4):** Librería de gráficos compatible con D3 v7.22
5. **SheetJS (xlsx.full.min.js):** Para la lectura del Excel.23

### 5.2 Estructura del Layout (Bootstrap Grid)

Se recomienda una distribución de tres niveles:

* **Nivel Superior (KPIs):** Tres tarjetas (Cards) mostrando métricas agregadas: "Total Incidentes", "Total Detenidos", "Incidentes Hoy". Utilizar dc.numberDisplay para estos valores.
* **Nivel Medio (Visualización Principal):**
  + Columna Izquierda (33%): Gráfico de Pastel (Tipo de Incidente).
  + Columna Central (33%): Gráfico de Filas (Cuadrantes).
  + Columna Derecha (33%): Gráfico de Filas (Oficiales).
* **Nivel Inferior (Detalle Temporal y Tabular):**
  + Fila Completa: Gráfico de Barras (Cronología) con selector de rango (range chart).
  + Fila Completa: Tabla de Datos (dc.dataTable) detallada.

### 5.3 Lógica de Procesamiento (JavaScript)

El siguiente pseudocódigo técnico describe la lógica crítica para procesar el archivo de CJB:

JavaScript

// Configuración del Parser de Fecha para formato DD/MM/YYYY HH:MM  
const parseDate = d3.timeParse("%d/%m/%Y %H:%M");  
  
// Función de carga activada por el input file  
function handleFileSelect(evt) {  
 const file = evt.target.files;  
 const reader = new FileReader();  
   
 reader.onload = function(e) {  
 const data = new Uint8Array(e.target.result);  
 const workbook = XLSX.read(data, {type: 'array'});  
   
 // Convertir primera hoja a JSON  
 const json\_data = XLSX.utils.sheet\_to\_json(workbook.Sheets]);  
   
 // Pipeline de Limpieza  
 json\_data.forEach(d => {  
 // Parsing de Fecha  
 d.dt = parseDate(d['Hora de inicio']);   
 // Si falla el parseo (fecha excel serial), intentar conversión numérica  
 if (d.dt == null && typeof d['Hora de inicio'] === 'number') {  
 d.dt = new Date(Math.round((d['Hora de inicio'] - 25569)\*86400\*1000));  
 }  
   
 // Normalización de Cuadrante  
 d.cuadrante = d['Cuadrante donde sucedió el hecho']? d['Cuadrante donde sucedió el hecho'].trim().toUpperCase() : 'DESCONOCIDO';  
   
 // Conteo de Involucrados (Iterar columnas Persona1...4)  
 let count = 0;  
 if(d['Nombre Completo']) count++;  
 if(d['Nombre Completo1']) count++;  
 if(d['Nombre Completo2']) count++;  
 if(d['Nombre Completo3']) count++;  
 if(d['Nombre Completo4']) count++;  
 d.total\_involucrados = count;  
 });  
  
 // Iniciar Dashboard  
 initDashboard(json\_data);  
 };  
 reader.readAsArrayBuffer(file);  
}  
  
function initDashboard(data) {  
 const ndx = crossfilter(data);  
 const all = ndx.groupAll();  
  
 // Definición de Dimensiones  
 const dimType = ndx.dimension(d => d |  
  
| 'Sin Clasificar');  
 const dimQuadrant = ndx.dimension(d => d.cuadrante);  
 const dimTime = ndx.dimension(d => d.dt);  
 const dimOfficer = ndx.dimension(d => d['Oficial a cargo1'] |  
  
| 'Sin Asignar');  
  
 // Definición de Grupos (Reducción)  
 const groupType = dimType.group();  
 const groupQuadrant = dimQuadrant.group();  
 const groupTime = dimTime.group(d3.timeDay); // Agrupar por día  
   
 // Configuración de Gráficos dc.js  
 //... (Código de configuración de charts aquí)  
   
 dc.renderAll();  
}

## 6. Recomendaciones Estratégicas para la Dirección Social

La implementación de este dashboard habilita nuevas capacidades de gestión para la Dirección Social de CJB.

1. Detección Temprana de Desplazamiento del Delito:

Al monitorear el gráfico de cuadrantes semanalmente, la Dirección podrá observar si las intervenciones en un sector (ej. B1) están causando un "efecto globo", desplazando la actividad delictiva o las infracciones hacia cuadrantes adyacentes (ej. B2). Esto permite una respuesta dinámica, ajustando los perímetros de seguridad antes de que el crimen se asiente en nuevas zonas.

2. Optimización de la Seguridad Municipal:

El análisis cruzado de "Seguridad Municipal" con "Hora de Inicio" permitirá identificar patrones en violaciones de normas de convivencia (ruido, uso indebido de espacios). Si estos incidentes se concentran los fines de semana en la noche, la Dirección puede programar personal de Seguridad Municipal específicamente para esos bloques horarios, optimizando el presupuesto de horas extra.

3. Transparencia y Rendición de Cuentas:

La visualización del desempeño por oficial y la tabla detallada de "Acciones Tomadas" proveen una base auditada para la evaluación del personal. Se recomienda establecer revisiones mensuales basadas en los datos del dashboard para identificar necesidades de capacitación en procedimientos de arresto o manejo de crisis, basándose en la narrativa de los incidentes más frecuentes.

## Conclusión

La transformación del formulario de Microsoft Forms en un sistema de inteligencia visual representa un paso decisivo hacia la modernización de la seguridad en Ciudad Juan Bosch. La solución propuesta, basada en **dc.js y Crossfilter**, cumple con todos los requisitos funcionales: es dinámica, interactiva, no requiere costos de licencia adicionales y mantiene la soberanía de los datos al operar localmente.

Esta arquitectura convierte datos administrativos inertes en una herramienta de conocimiento situacional. Al permitir que los tomadores de decisiones filtren por cuadrante, tipo de incidente y oficial con un simple clic, se reduce el tiempo de análisis de horas a segundos. La capacidad de visualizar la compleja realidad operativa de CJB —con sus múltiples actores y tipologías de incidentes— en un único panel de control coherente, empoderará a la Dirección Social para actuar con precisión quirúrgica, mejorando la seguridad y la calidad de vida de las 25,000 familias residentes.

### Apéndice: Mapeo de Columnas para el Dashboard

| **Campo Dashboard** | **Columna Excel Origen** | **Transformación Requerida** |
| --- | --- | --- |
| **Filtro de Fecha** | Hora de inicio | Parseo con d3.timeParse a objeto Date JS. |
| **Selector de Cuadrante** | Cuadrante donde sucedió el hecho | Normalización a Mayúsculas, manejo de nulos. |
| **Categoría de Incidente** | Tipo de Incidente | Fusión condicional con Incidentes relacionados a tránsito si aplica. |
| **Desempeño Oficial** | Oficial a cargo1 | Agrupación y conteo simple. |
| **Detalle de Acción** | Acciones Tomadas | Separación por delimitador (split by ';') para análisis de frecuencia. |
| **Impacto Humano** | Rol de la Persona (1-4) | Algoritmo de conteo iterativo para sumar total de involucrados. |
| **Evidencia** | Evidencia Visual | Generación de etiqueta HTML <a href="..."> en la tabla de detalles. |

#### Fuentes citadas

1. Registro Rápido de Incidentes (SEGURIDAD).csv
2. d3-time-format | D3 by Observable - D3.js, acceso: diciembre 15, 2025, <https://d3js.org/d3-time-format>
3. Working With Dates - Using D3.js, acceso: diciembre 15, 2025, <http://using-d3js.com/04_04_working_with_dates.html>
4. Crossfilter, acceso: diciembre 15, 2025, <https://square.github.io/crossfilter/>
5. Create an Interactive Dashboard with Crossfilter and Dc.Js | by Louis Nicolle | Medium, acceso: diciembre 15, 2025, <https://medium.com/@louisn_23157/interactive-dashboard-crossfilter-dcjs-tutorial-7f3a3ea584c2>
6. Nasdaq 100 Index 1985/11/01-2012/06/29 - dc.js - Dimensional Charting Javascript Library, acceso: diciembre 15, 2025, <https://dc-js.github.io/dc.js/>
7. How to Read and Write Excel file in Node.js ? - GeeksforGeeks, acceso: diciembre 15, 2025, <https://www.geeksforgeeks.org/node-js/how-to-read-and-write-excel-file-in-node-js/>
8. Get browser data into Excel with SheetJS | Berg Software, acceso: diciembre 15, 2025, <https://www.berg-software.com/en/get-browser-data-into-excel-with-sheetjs/>
9. Free Bootstrap Admin Template · Open Source Dashboard & Panel - CoreUI, acceso: diciembre 15, 2025, <https://coreui.io/product/free-bootstrap-admin-template/>
10. Responsive Visualizations Using D3.js and Bootstrap - Packt, acceso: diciembre 15, 2025, <https://www.packtpub.com/en-us/learning/how-to-tutorials/responsive-visualizations-using-d3js-and-bootstrap>
11. Getting to know Crossfilter - Create With Data, acceso: diciembre 15, 2025, <https://www.createwithdata.com/articles/crossfilter/>
12. How to Create Charts from External Data Sources with D3.js - Bits and Pieces, acceso: diciembre 15, 2025, <https://blog.bitsrc.io/how-to-create-charts-from-external-data-sources-with-d3-js-4abbcb574706>
13. Source: charts/row-chart.js, acceso: diciembre 15, 2025, <https://dc-js.github.io/dc.js/docs/html/charts_row-chart.js.html>
14. dc.js Getting Started and How-To Guide - dc.js - Dimensional Charting Javascript Library, acceso: diciembre 15, 2025, <https://dc-js.github.io/dc.js/docs/stock.html>
15. DC.js - Data Table - Tutorials Point, acceso: diciembre 15, 2025, <https://www.tutorialspoint.com/dcjs/dcjs_data_table.htm>
16. Class: dataTable - dc.js - Dimensional Charting Javascript Library, acceso: diciembre 15, 2025, <https://dc-js.github.io/dc.js/docs/html/dc.dataTable.html>
17. Need to display crossfilter data array values in separate lines instead of comma separated values in a single line - Stack Overflow, acceso: diciembre 15, 2025, <https://stackoverflow.com/questions/55581244/need-to-display-crossfilter-data-array-values-in-separate-lines-instead-of-comma>
18. How to split comma, semicolon and comma separated phrases with semicolons around?, acceso: diciembre 15, 2025, <https://stackoverflow.com/questions/55403175/how-to-split-comma-semicolon-and-comma-separated-phrases-with-semicolons-around>
19. Source: data-table.js - dc.js - Dimensional Charting Javascript Library, acceso: diciembre 15, 2025, <https://dc-js.github.io/dc.js/docs/html/data-table.js.html>
20. Getting started | D3 by Observable - D3.js, acceso: diciembre 15, 2025, <https://d3js.org/getting-started>
21. crossfilter/crossfilter: Fast n-dimensional filtering and grouping of records. - GitHub, acceso: diciembre 15, 2025, <https://github.com/crossfilter/crossfilter>
22. dc-v4-upgrade-guide.md - GitHub, acceso: diciembre 15, 2025, <https://github.com/dc-js/dc.js/blob/develop/docs/dc-v4-upgrade-guide.md>
23. xlsx - NPM, acceso: diciembre 15, 2025, <https://www.npmjs.com/package/xlsx>