

# Hackeando el Sistema Legal Chileno

Documentación técnica v1.0 | Onboarding para Developers

## System Info

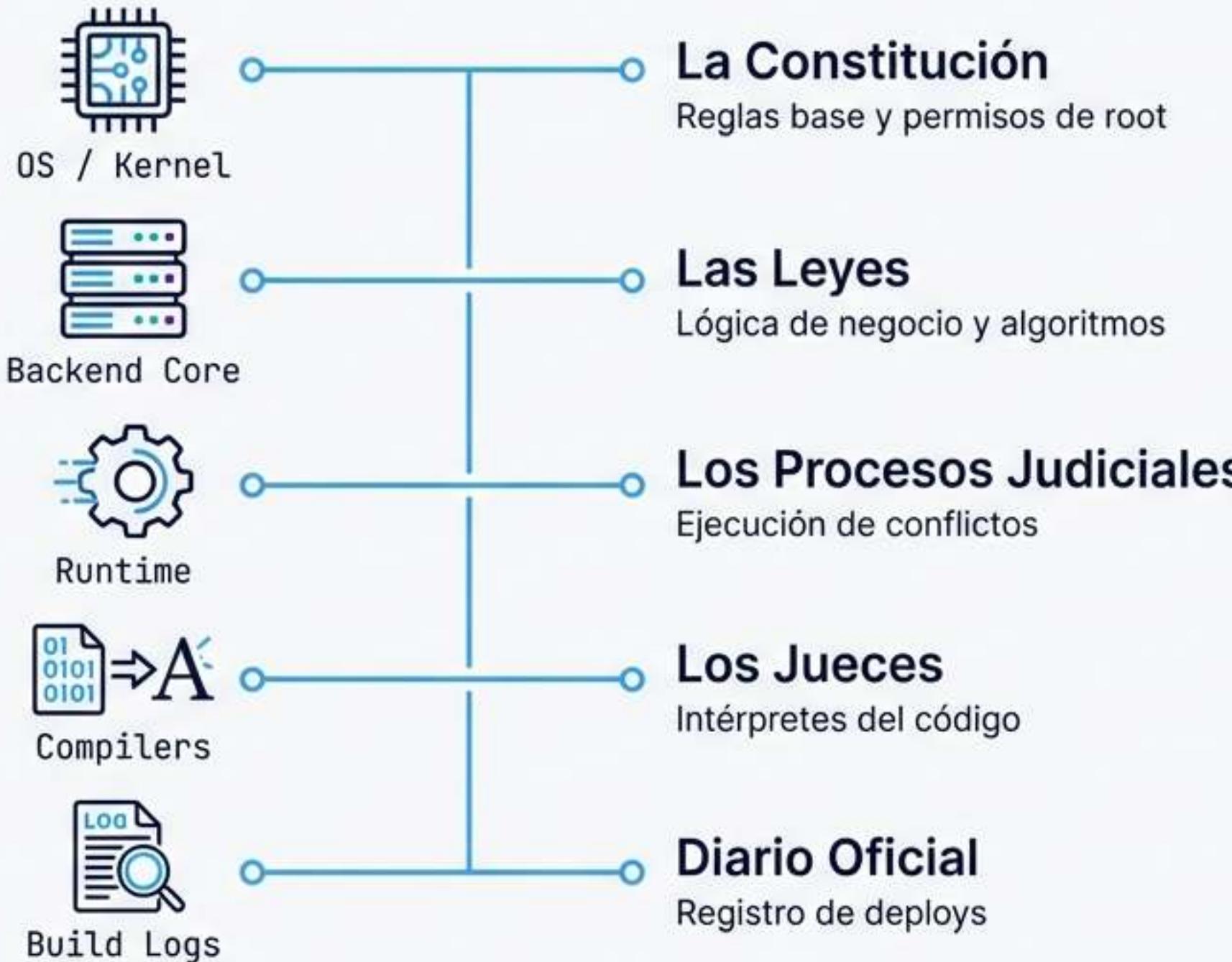
// System Status

- > **Context:** El derecho no es magia; es un sistema operativo con reglas, permisos, usuarios y protocolos.
- > **Objective:** Desmitificar la abstracción legal utilizando lógica de programación.
- > **Environment:** Stack Chile (Civil Law System)



LEGAL PROCESSING UNIT

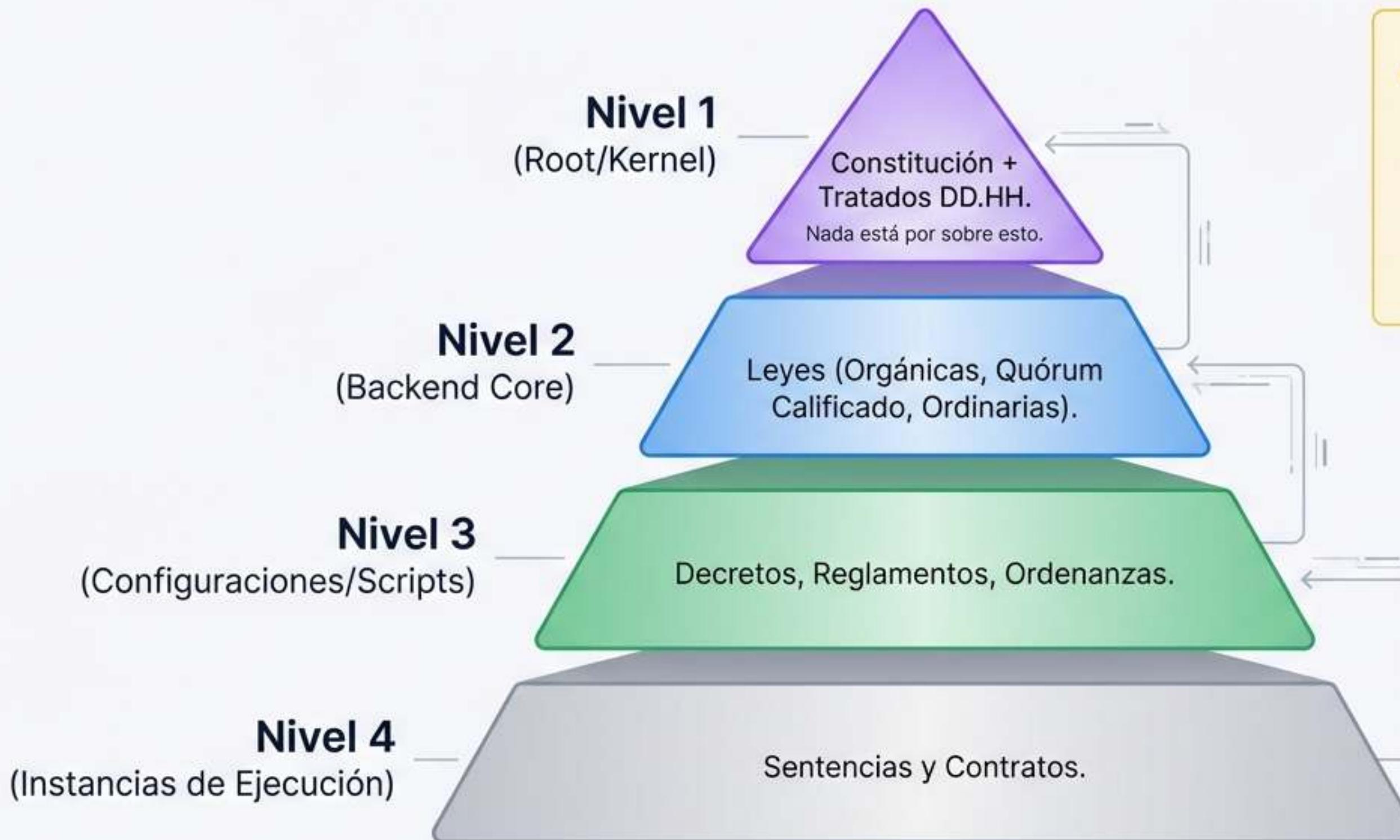
# El Modelo Mental: The Full Stack Overview



## Insight

Entender el sistema legal requiere dejar de memorizar textos y empezar a entender la arquitectura de flujo de datos.

# Jerarquía del Sistema (Root Access)



## Error de Jerarquía:

Si un Reglamento (Nivel 3) contradice a una Ley (Nivel 2), el sistema aplica la Ley.

Una norma inferior nunca puede sobrescribir permisos de Root.

# Configuración de Permisos: Pùblico vs. Privado



**\*\*Default Deny\*\***

## Derecho Pùblico

Estado vs. Ciudadano

Principio de Legalidad: El Estado solo puede hacer lo que la ley explícitamente permite.

```
if (!action.isExplicitlyAllowed()) {  
    return 'Prohibido';  
}
```



**\*\*Default Allow\*\***

## Derecho Privado

Ciudadano vs. Ciudadano

Autonomía de la Voluntad: Las personas pueden hacer todo lo que la ley no prohíba explícitamente.

```
if (!action.isExplicitlyForbidden()) {  
    return 'Permitido';  
}
```

# The Build Pipeline: Ciclo de Vida de una Ley



// NOTE: Una ley no existe hasta el Deploy final en el Diario Oficial.  
Antes de eso, es solo código en desarrollo.

# Repositories & Version Control (Data Sources)



## Diario Oficial

The Build Log

Registro inmutable y atómico.  
Foto estática del día del deploy.

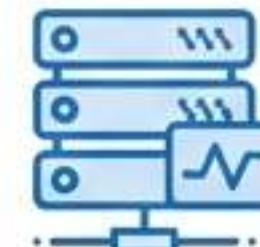


## Biblioteca del Congreso (BCN)

Git Repo / Docs

Fuente de verdad actualizada.  
Ley compilada con todos los  
parches aplicados.

ⓘ Usar BCN es leer la  
documentación oficial.



## Poder Judicial

Runtime Logs

Historial de casos y sentencias.

- ⚙️ **Tech Concept: Vigencia Diferida = Feature Flags.** Una ley publicada hoy puede activarse en 6 meses.
- ⚠️ **Warning:** Usar Google es como copiar código de un foro de 2018 (probablemente depreciado).



# Protocolos de Ejecución (Tipos de Procesos)

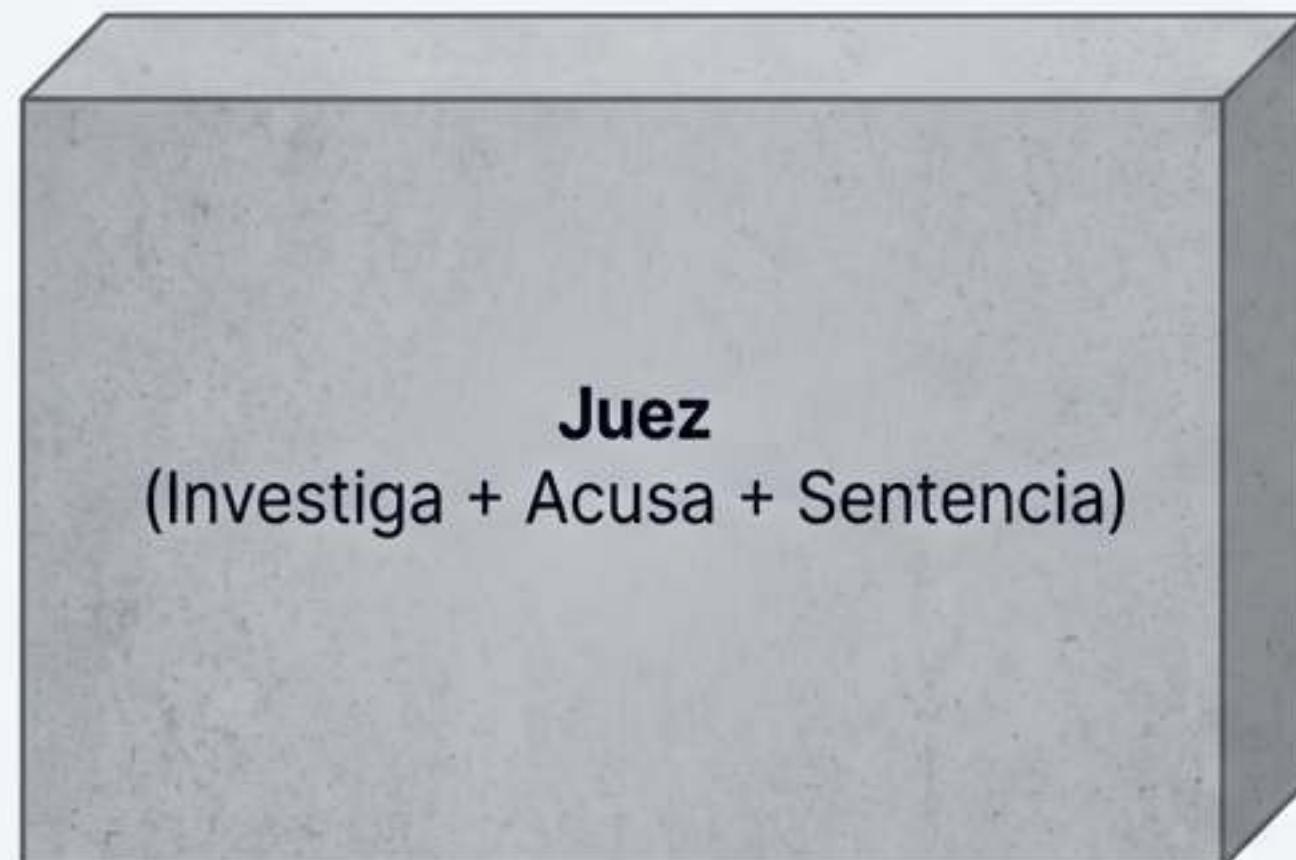


// EDGE CASES: Policía Local (Tránsito) y Administrativos y Multas SII.

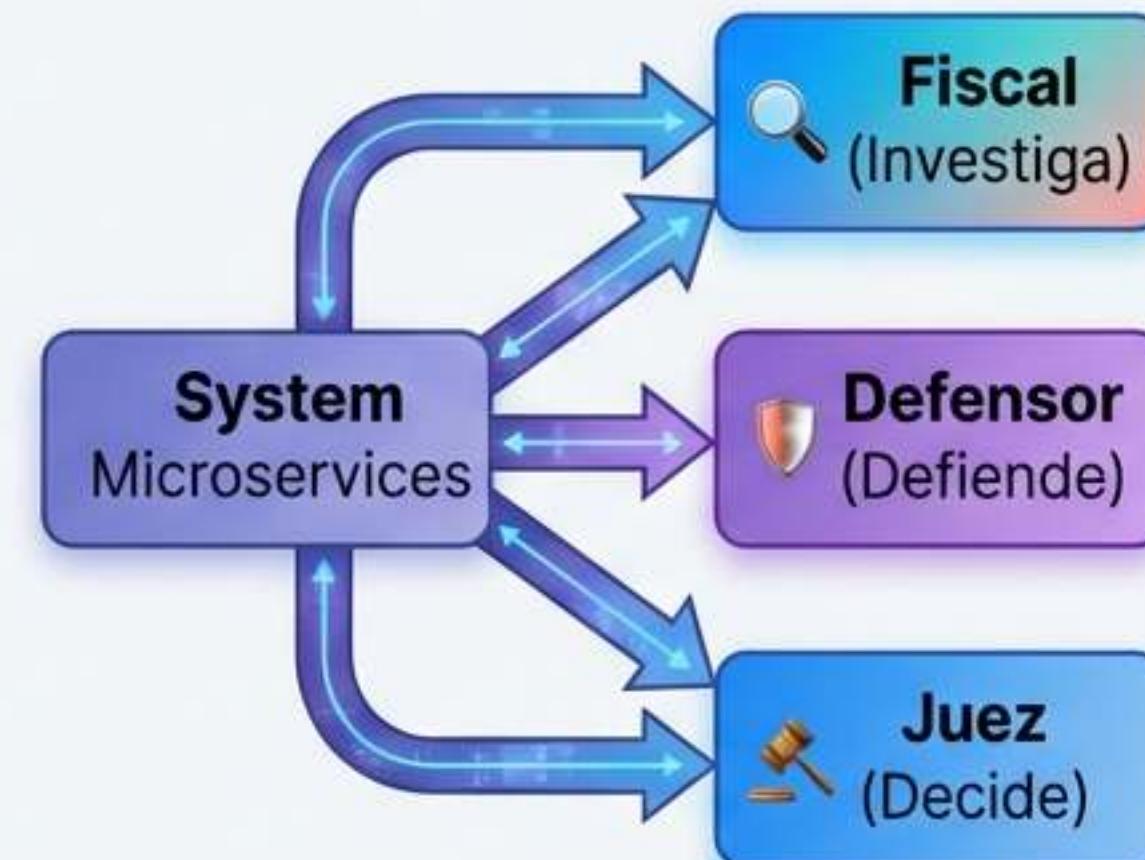
# System Update: Legacy vs. Modern Architecture

## Refactoring the Judiciary

### Legacy (Sistema Inquisitivo - Pre 2005)



### Modern (Sistema Acusatorio)



**Architecture:** Monolítico/Opaco.

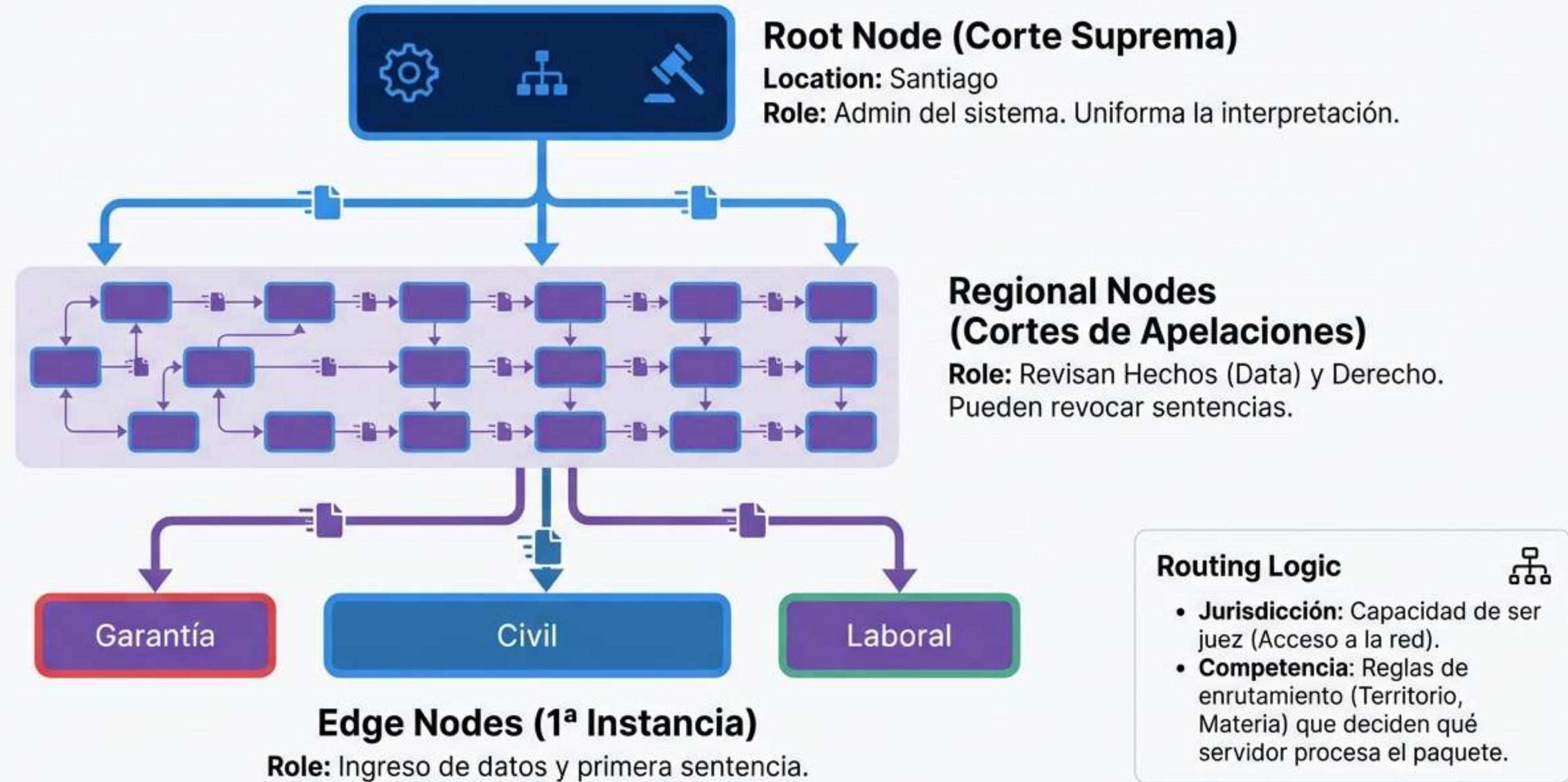
**⚠ Bug:** Un solo juez hace todo. Alto sesgo. Proceso escrito y lento.

**Architecture:** Separation of Concerns.

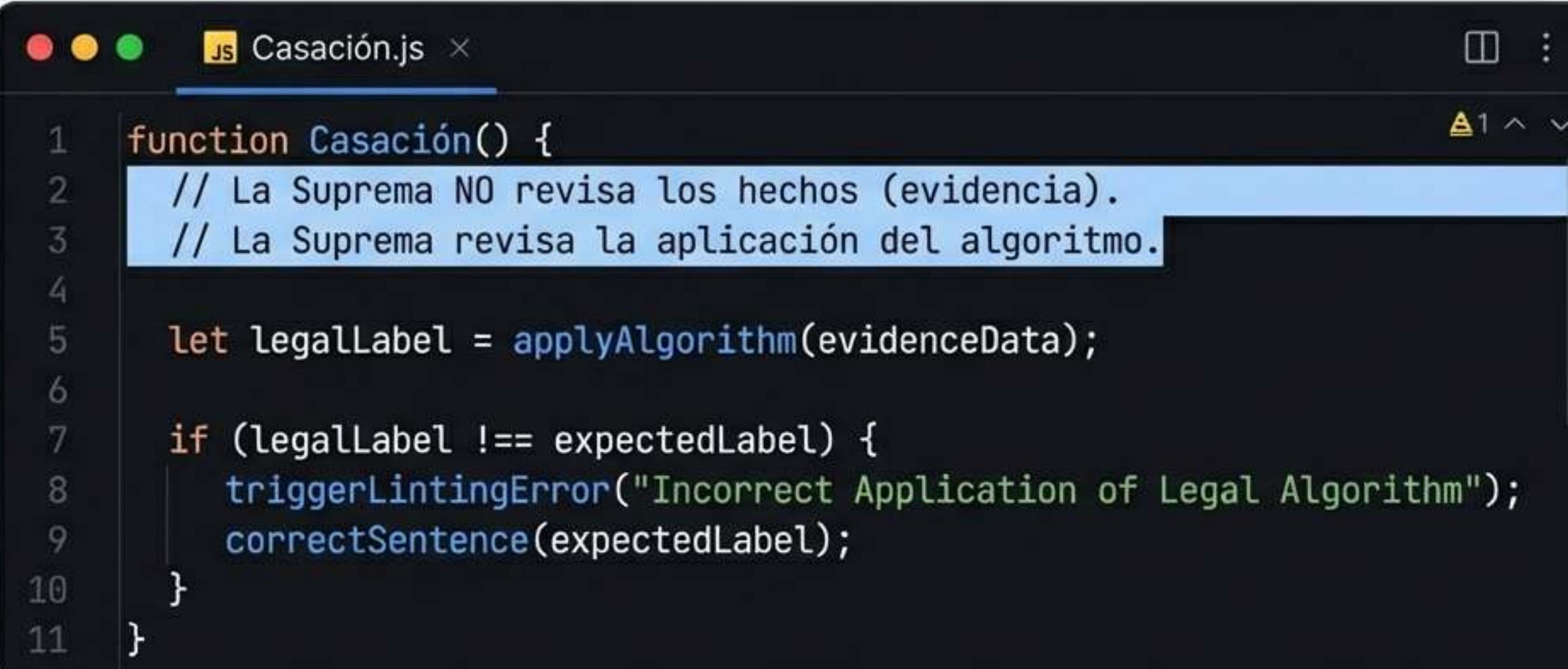
**✓ Feature:** Inmediación. El juez debe ver la prueba en vivo (Oralidad). Acceso a Raw Data vs. Compressed Data.

**Tech Concept:** The shift from a Monolithic structure (Sistema Inquisitivo) to a Microservices architecture (Sistema Acusatorio) isolates roles, reduces bias, and improves data throughput (Oralidad).

# Network Topology (Estructura de Tribunales)



# Debugging & Static Analysis (Corte Suprema)



```
function Casación() {
    // La Suprema NO revisa los hechos (evidencia).
    // La Suprema revisa la aplicación del algoritmo.

    let legalLabel = applyAlgorithm(evidenceData);

    if (legalLabel !== expectedLabel) {
        triggerLintingError("Incorrect Application of Legal Algorithm");
        correctSentence(expectedLabel);
    }
}
```

## The Bug Fix



**The Bug:** El tribunal inferior estableció bien los hechos, pero aplicó la etiqueta 'Homicidio Simple' en lugar de 'Asesinato'.



**The Fix:** La Suprema corrige la sentencia para asegurar que el algoritmo legal corra igual en todo el país.

# User Classes: Perfiles de Abogados



## Corporativo

(Dev / Architect)

- **Goal:** Prevenir bugs (conflictos).
- **Task:** Diseño de contratos y estructuras (M&A, Compliance).
- **Mindset:** 'Si vamos a juicio, el diseño falló.'



## Litigante

(Ops / Incident Response)

- **Goal:** Resolver el conflicto / Minimizar daños.
- **Task:** 'Front-end' del tribunal. Estrategia de prueba.
- **Mindset:** 'Destruir el argumento de la contraparte.'

# Authentication & Integrity (Terceros de Confianza)

## Notario (Certificate Authority - CA)

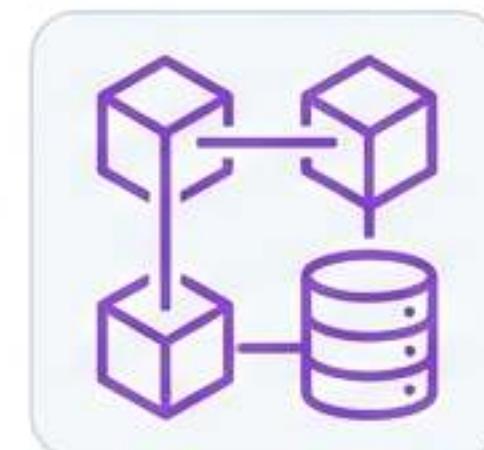


Ministro de fe. Validador de identidad y timestamp.

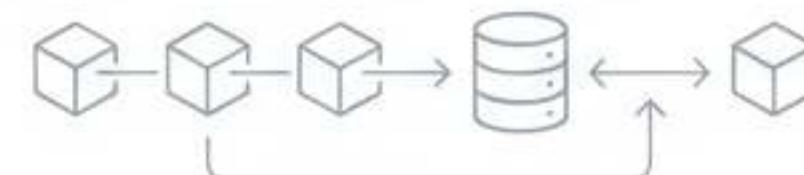


Analogía: "Doy fe que User X firmó el día Y". Evita el repudio de firma.

## Conservador de Bienes Raíces - CBR (Immutable Ledger)



Base de datos transaccional maestra.

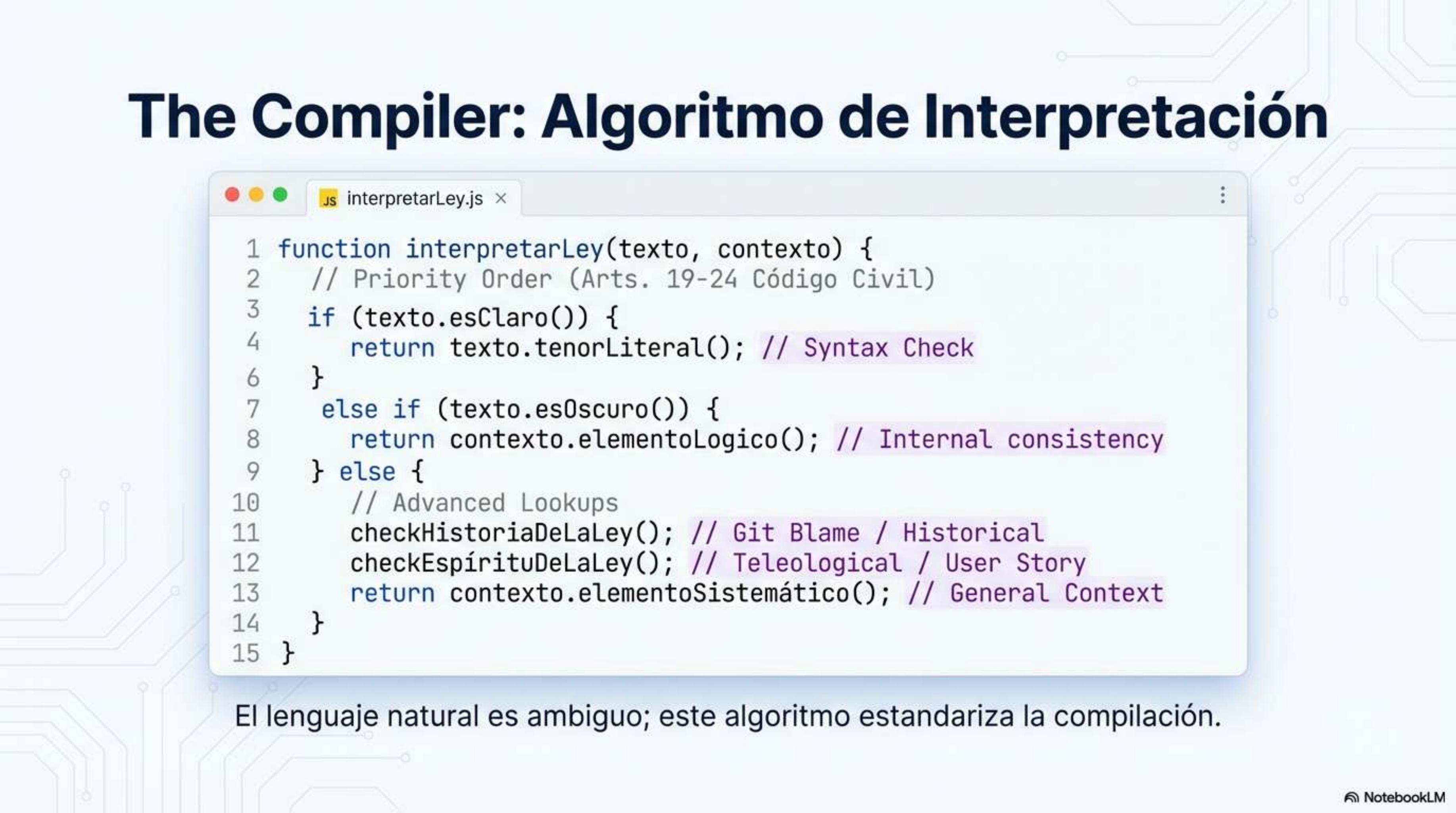


Prevention: Evita el "Double Spending" (Vender la misma casa a dos personas). No eres dueño hasta que te inscribes en el Ledger.

# System Defenders: Roles del Estado



# The Compiler: Algoritmo de Interpretación



```
● ○ ● JS interpretarLey.js × :  
1 function interpretarLey(texto, contexto) {  
2     // Priority Order (Arts. 19-24 Código Civil)  
3     if (texto.esClaro()) {  
4         return texto.tenorLiteral(); // Syntax Check  
5     }  
6     else if (texto.esOscuro()) {  
7         return contexto.elementoLogico(); // Internal consistency  
8     } else {  
9         // Advanced Lookups  
10        checkHistoriaDeLaLey(); // Git Blame / Historical  
11        checkEspírituDeLaLey(); // Teleological / User Story  
12        return contexto.elementoSistemático(); // General Context  
13    }  
14 }  
15 }
```

El lenguaje natural es ambiguo; este algoritmo estandariza la compilación.

# System Logs & Pattern Recognition (Jurisprudencia)

Civil Law (Chile)	Common Law (USA/UK)
<b>Rule:</b> Efecto relativo de las sentencias.	<b>Rule:</b> Stare Decisis.
<b>Status:</b> La ley es la fuente de verdad. El fallo anterior es un <i>Log</i> referencial. 	<b>Status:</b> El caso anterior *ES* la ley. 
<b>Binding? NO.</b> (Un juez puede cambiar de opinión).	<b>Binding? SÍ.</b>

Takeaway: En Chile, usamos la jurisprudencia como '**Business Intelligence**' para predecir resultados, no como código duro.

