Informe Técnico-Jurídico Integral: Desarrollo de Plataforma de Gestión de Reclamaciones para el Cártel de Fabricantes de Automóviles (2006-2013) bajo Arquitectura Vue.js 3

1. Introducción y Definición del Alcance del Proyecto

El presente informe constituye un análisis exhaustivo y pormenorizado del proyecto de desarrollo de software denominado "Aplicación de Gestión Cártel Coches", concebido como una solución tecnológica para abordar una de las mayores crisis de competencia en la historia industrial de España. Este documento tiene como objetivo fusionar dos dominios de conocimiento complejos: la ingeniería de software moderna —específicamente el ecosistema Vue.js 3— y el marco jurídico-económico derivado de las sanciones impuestas por la Comisión Nacional de los Mercados y la Competencia (CNMC) a los fabricantes de automóviles que operaron en el mercado español entre los años 2006 y 2013.

El proyecto, tal como se define en la documentación técnica preliminar [1], busca implementar una *Single Page Application* (SPA) que permita el registro, gestión y seguimiento de reclamaciones de afectados. Sin embargo, la aparente sencillez de un "sistema de gestión" esconde una complejidad subyacente crítica: la necesidad de modelar reglas de negocio legales extremadamente volátiles y específicas, donde la viabilidad de una reclamación depende de la intersección precisa entre la marca del vehículo, la fecha de compra y la sentencia específica aplicable a dicho fabricante.[2]

En el contexto actual de 2026, la relevancia de esta herramienta se ha visto magnificada por las recientes sentencias del Tribunal de Justicia de la Unión Europea (TJUE) y el Tribunal Supremo, que han extendido los plazos de prescripción mucho más allá de las previsiones iniciales, permitiendo reclamaciones activas hasta bien entrado el año 2026.[4] Por tanto, la aplicación no es un mero ejercicio académico, sino un prototipo de herramienta *LegalTech* con impacto social directo sobre millones de consumidores que pagaron sobrecostes estimados entre el 10% y el 15% del valor de sus vehículos.[6]

Este informe desglosará, en primera instancia, la fenomenología del Cártel de Coches para establecer las "Reglas de Negocio"; posteriormente, diseccionará la arquitectura técnica propuesta en Vue.js 3, justificando las decisiones de diseño; y finalmente, abordará la implementación crítica de la validación de datos y la experiencia de usuario, elementos

indispensables para la eficacia jurídica de la herramienta.

---

2. Análisis del Dominio Legal: El Cártel de Fabricantes (2006-2013)

Para que el equipo de desarrollo pueda construir una aplicación robusta, es imperativo comprender la naturaleza de los datos que se procesarán. El término "Cártel de Coches" hace referencia a una infracción continuada del artículo 1 de la Ley 15/2007 de Defensa de la Competencia y del artículo 101 del Tratado de Funcionamiento de la Unión Europea. No se trató de un evento aislado, sino de una práctica sistémica de intercambio de información confidencial.

2.1. Naturaleza de la Infracción y Mecánica del Fraude

La infracción, sancionada por la CNMC en su resolución de julio de 2015, consistió en el intercambio secreto, futuro y estratégico de información entre las marcas competidoras. Este intercambio abarcaba tres áreas fundamentales que la aplicación debe considerar al evaluar la documentación aportada por el usuario:

1. **Gestión Empresarial:** Intercambio de datos sobre rentabilidad de la red de concesionarios.
2. **Postventa:** Unificación de criterios y precios en servicios de taller y recambios.
3. **Marketing:** Conocimiento previo de las campañas promocionales de la competencia.

El efecto económico de esta colusión fue la eliminación de la incertidumbre en el mercado. Al saber de antemano las estrategias de los rivales, los fabricantes pudieron mantener los precios artificialmente altos, eliminando los descuentos agresivos que caracterizan a un mercado de libre competencia. Los estudios econométricos presentados en los litigios estiman que este sobreprecio osciló entre un 10% y un 15% del valor de factura del vehículo antes de impuestos.[6]

Implicación para la Base de Datos: La aplicación no puede tratar todas las reclamaciones por igual. El daño (cuantía a reclamar) varía según el modelo econométrico aplicable a cada marca. Aunque la aplicación descrita es de "gestión" y no de "cálculo pericial", es vital que la estructura de datos (JSON en Google Sheets) contemple campos para almacenar el "Precio Base" (sin IVA) y la "Fecha de Matriculación", variables críticas para el algoritmo de indemnización.[8]

2.2. Cronología y Plazos de Prescripción: La Ventana de Oportunidad 2026

Uno de los desafíos más grandes para el desarrollo del módulo de validación (ClienteForm.vue

y VehiculoForm.vue) es la determinación de si una reclamación está "en plazo".

Históricamente, el plazo para reclamar daños extracontractuales en España es de un año (Art. 1968 Código Civil). Sin embargo, dada la ocultación de la conducta (el cártel era secreto), el TJUE ha dictaminado que el plazo ("dies a quo") no comienza hasta que el perjudicado tiene un conocimiento razonable de la infracción y de que puede reclamar. Esto, sumado a la Directiva de Daños de la UE que establece un plazo de 5 años, y las interrupciones por la pandemia COVID-19, ha desplazado la fecha límite.

La investigación confirma que, para la mayoría de las marcas, el plazo sigue abierto en **2026**:

- **Sentencias del Tribunal Supremo (2021):** La mayoría de las ratificaciones de multas ocurrieron en 2021. Sumando 5 años desde esa fecha, nos situamos en 2026.
- **Interrupción de la Prescripción:** Cualquier reclamación extrajudicial previa interrumpe el plazo, reiniciando la cuenta de 5 años. La aplicación debe incluir un campo para registrar si el usuario ha enviado ya una "Reclamación Previa" al fabricante, ya que esto valida legalmente reclamaciones que de otro modo estarían prescritas.[9]

2.3. Matriz de Marcas Afectadas y Periodos de Infracción

El corazón lógico de la aplicación reside en la validación cruzada entre Marca y Fecha de Compra. A diferencia de una validación simple, aquí nos enfrentamos a una lógica condicional compleja. No basta con validar que la fecha esté entre 2006 y 2013; cada marca tiene su propia "ventana de infracción" definida por la sentencia específica que la condenó.

A continuación, se presenta la matriz de datos que debe alimentar el componente SelectField.vue (o BaseSelect.vue) para la selección de marcas, basada en las resoluciones de la CNMC y las sentencias firmes.[2] Esta tabla es fundamental para configurar las reglas de validación en el *frontend*:

Grupo Automotriz	Marca Comercial	Inicio Infracción	Fin Infracción	Notas Técnicas para Desarrollo
Volkswagen Group	Audi	Feb 2006	Jul 2013	Periodo estándar completo.
	SEAT	Feb 2006	Jul 2013	Solicitó clemencia (delator), pero es demandable

				por daños a particulares.
	Volkswagen (VW)	Feb 2006	Jul 2013	
	Skoda	Feb 2006	Jul 2013	
PSA (Stellantis)	Citroën	Feb 2006	Jul 2013	Alto volumen de ventas (C4, Xsara Picasso).
	Peugeot	Feb 2006	Jul 2013	
Renault	Renault	Feb 2006	Jul 2013	Incluye la red comercial propia.
General Motors	Opel	Feb 2006	Jul 2013	General Motors España.
	Chevrolet	Feb 2006	Jul 2013	Marca ya no opera en Europa; notificaciones complejas.
Ford	Ford	Feb 2006	Jul 2013	
Fiat (FCA)	Fiat	Feb 2006	Jul 2013	
	Alfa Romeo	Feb 2006	Jul 2013	
	Lancia	Feb 2006	Jul 2013	Marca residual, bajo volumen.

Toyota	Toyota	Feb 2006	Jul 2013	
	Lexus	Feb 2006	Jul 2013	Segmento Premium.
Nissan	Nissan	Jun 2008	Ago 2013	**Atención:** Inicio posterior (2008). Validar fechas < 2008 como inválidas.
BMW	BMW	Jun 2008	Nov 2009	**Ventana Crítica:** Solo 17 meses. Alto riesgo de rechazo.
	Mini	Jun 2008	Nov 2009	Misma ventana reducida que BMW.
Honda	Honda	Abr 2009	Abr 2012	**Ventana Reducida.**
Mercedes-Benz	Mercedes	Mar 2010	Feb 2011	**Ventana Extremadamente Corta (11 meses).**
Volvo	Volvo	Mar 2010	Feb 2011	**Ventana Extremadamente Corta.**
Mitsubishi	Mitsubishi	Mar 2010	Ago 2013	Inicio tardío.
Mazda	Mazda	Mar 2010	Feb 2012	Periodo intermedio.

Hyundai	Hyundai	Jul 2011	Jul 2013	Solo final del periodo.
Kia	Kia	Mar 2007	Nov 2012	Periodo irregular.

Análisis de Implicaciones Técnicas:

Esta heterogeneidad en las fechas obliga a que el componente VehiculoForm.vue no sea estático. Se debe implementar un patrón de **validación asimétrica**:

1. El usuario selecciona la "Marca".
2. El sistema consulta la matriz de fechas (almacenada en un archivo constants.js o recuperada del Google Sheet).
3. El *datepicker* o selector de fecha restringe las opciones seleccionables, o bien, el sistema emite un *warning* visual si la fecha introducida está fuera del rango sancionado para esa marca específica.
4. Esto evita que la aplicación registre reclamaciones inviables (falsos positivos), mejorando la calidad de los datos para el administrador.[1]

---

3. Arquitectura Técnica: Implementación en Vue.js 3

El proyecto se define técnicamente como una *Single Page Application* (SPA) construida sobre el *framework* progresivo Vue.js en su versión 3. Esta elección tecnológica es idónea para los requisitos planteados, dado que Vue 3 ofrece un rendimiento superior y una mejor gestión de la lógica compleja en comparación con su predecesor, gracias a la *Composition API*.

3.1. Ecosistema y Herramientas de Construcción (Build Tools)

El documento de requisitos [1] especifica el uso de **Vite** como herramienta de construcción. Esta elección es estratégica frente a Webpack. Vite utiliza el soporte nativo de los navegadores para módulos ES (ECMAScript Modules), lo que permite un inicio del servidor de desarrollo casi instantáneo y un *Hot Module Replacement* (HMR) extremadamente rápido. En un proyecto académico o profesional ágil, esto reduce drásticamente los tiempos de espera durante la codificación.

Estructura del Proyecto Propuesta:

Siguiendo las mejores prácticas de Vue 3, la estructura de directorios debe ser modular y escalable:

```
src/
├── assets/            # Estilos globales (CSS/SASS), imágenes, fuentes
├── components/        # Componentes reutilizables (Átomos y Moléculas)
│   ├── ui/            # UI Kit base (BaseInput, BaseSelect, BaseButton)
│   ├── forms/         # Formularios complejos (ClienteForm, VehiculoForm)
│   └── layout/        # Header, Footer, Sidebar
├── composables/       # Lógica reutilizable (useDniValidation, useGoogleSheets)
├── router/            # Configuración de Vue Router (index.js)
├── stores/            # Gestión de estado global (Pinia o Reactive Store)
├── utils/             # Funciones auxiliares (validadores regex, formateadores)
├── views/             # Páginas principales (Home, AdminPanel)
├── App.vue            # Componente raíz
└── main.js            # Punto de entrada
```

3.2. Gestión de Estado y Lógica Reactiva (Composition API)

El requisito de usar ref, reactive y computed [1] apunta directamente al uso de la **Composition API**. A diferencia de la *Options API* (basada en objetos data, methods, mounted), la Composition API permite agrupar la lógica por "funcionalidad" (feature) en lugar de por "tipo de opción".

Para el formulario de registro de vehículos, que posee una alta complejidad lógica (dependencias entre campos), el uso de <script setup> simplifica el código:

JavaScript

```
<script setup>
import { ref, computed, watch } from 'vue';
import { marcas } from '@/utils/constants'; // Matriz de marcas y fechas

// Estado reactivo del formulario
const form = reactive({
  marca: '',
  modelo: '',
  fechaMatriculacion: ''
});

// Lógica de Modelos Dependientes
```

```
const modelosDisponibles = ref();

// Watcher: Cuando cambia la marca, actualizamos los modelos
watch(() => form.marca, (nuevaMarca) => {
  if (nuevaMarca) {
    // Lógica para cargar modelos (podría ser una llamada API o filtro local)
    modelosDisponibles.value = obtenerModelosPorMarca(nuevaMarca);
    form.modelo = ''; // Reset del modelo al cambiar marca
  }
});

// Validación Computada en tiempo real
const esFechaValida = computed(() => {
  if (!form.marca ||!form.fechaMatriculacion) return true; // No validar si está vacío
  const rango = obtenerRangoMarca(form.marca);
  const fecha = new Date(form.fechaMatriculacion);
  return fecha >= rango.inicio && fecha <= rango.fin;
});
</script>
```

Esta estructura permite que la lógica de validación del cártel (que es compleja y varía por marca) se encapsule en funciones importables (obtenerRangoMarca), manteniendo el componente limpio y testeable.

3.3. Enrutamiento y Navegación (Vue Router)

La aplicación debe gestionar flujos distintos para usuarios (afectados) y administradores. **Vue Router** es esencial aquí para implementar "Navigation Guards" (Guardias de Navegación).

- **Rutas Públicas:** /, /registro-cliente, /registro-vehiculo.
- **Rutas Privadas:** /admin/dashboard.

El sistema debe proteger la ruta /admin verificando una autenticación simple (dado que el backend es Google Sheets, esto puede ser un *hardcoded check* o un token básico almacenado en localStorage).

JavaScript

```
// router/index.js
const routes =;
```

4. Integridad de Datos: Algoritmos de Validación Avanzada

Dado que el propósito de la aplicación es generar registros válidos para una reclamación legal, la integridad de los datos de entrada es prioritaria. Un error tipográfico en el DNI o la matrícula invalidaría el registro.

4.1. Validación de Identidad (DNI/NIE)

El documento [1] exige validación por Regex y algoritmo. En España, el Documento Nacional de Identidad (DNI) y el Número de Identificación de Extranjero (NIE) siguen una lógica de **Módulo 23**.

Lógica del Algoritmo:

1. **Formato DNI:** 8 dígitos + Letra de Control.
2. **Formato NIE:** Letra (X, Y, Z) + 7 dígitos + Letra de Control.
3. **Normalización:** Para el cálculo, las letras iniciales del NIE se sustituyen: X→0, Y→1, Z→2.
4. **Operación:** Se divide el número resultante entre 23. El resto de la división (un número entre 0 y 22) se mapea a una tabla de letras estandarizada: TRWAGMYFPDXBNJZSQVHLCKE.

Implementación en la App:

No basta con verificar que el usuario introdujo 8 números y una letra. El sistema debe recalcular la letra esperada y compararla con la introducida. Si no coinciden, se debe bloquear el envío del formulario. Esto es crítico para asegurar que los datos enviados a la hoja de cálculo sean utilizables para contactar al cliente.

Regex de Filtrado Previo:

- DNI: /^[0-9]{8}[A-Z]$/
- NIE: /^[0-9]{7}[A-Z]$/

4.2. Validación de Matrículas: Historia y Técnica

La validación de matrículas es compleja debido a la coexistencia de dos sistemas durante la vida útil de los coches afectados (algunos podrían ser re-matriculaciones o transferencias).

1. **Sistema Nacional (Vigente desde Septiembre 2000):**
    - Formato: 0000 BBB. Cuatro números y tres letras.
    - **Regla de Exclusión:** No se utilizan las vocales (A, E, I, O, U) para evitar palabras

malsonantes (ej. PIS, ANO). Tampoco se usan la Ñ ni la Q (por confusión con N y O/0).
    - **Regex Robusto:** /^\d{4}\s?{3}$/
    - *Insight:* Si el usuario intenta introducir una vocal en la matrícula, el sistema debe impedirlo y mostrar un mensaje educativo: "Las matrículas modernas no contienen vocales".
2. **Sistema Provincial (1971 - 2000):**
    - Formato: M-1234-AB. Código provincia (1-2 letras) + 4 números + Código serie (1-2 letras).
    - Aunque el cártel empezó en 2006, un coche comprado en 2006 podría ser un "Km 0" o un stock antiguo matriculado previamente, aunque es poco probable para reclamaciones de "vehículo nuevo". Sin embargo, el sistema debería estar preparado para manejar este formato si se permiten reclamaciones de segunda mano (aunque el cártel afecta primariamente a la primera compra).

4.3. Validación de Modelos y Versiones (Selects Dependientes)

La selección del modelo es vital. Un usuario puede decir que tiene un "Seat" pero seleccionar un modelo "Panda" (que es Fiat, o Seat antiguo). La lista de modelos debe filtrarse dinámicamente.

Datos para la Base de Datos de Modelos (JSON):

Para cubrir el grueso de las reclamaciones (modelos más vendidos 2006-2013), el JSON de modelos debe incluir:

- **Renault:** Mégane II y III, Clio III, Scénic.
- **Citroën:** C4 (Líder ventas 2013), C3, Xsara Picasso.
- **Seat:** Ibiza (6L/6J), León (1P), Altea.
- **Volkswagen:** Golf (V/VI), Polo, Passat.
- **Ford:** Focus (Mk2/Mk3), Fiesta.
- **Opel:** Astra (H/J), Corsa (D).
- **Nissan:** Qashqai (J10) - *Nota:* Este modelo revolucionó el mercado en 2007 y habrá miles de reclamaciones.

---

5. Estrategia de Persistencia: Google Sheets como Backend Serverless

El proyecto propone una solución innovadora y de bajo coste: utilizar **Google Sheets** como base de datos.[1] Esta aproximación, conocida como "No-Code Backend" o "Serverless Sheets", es ideal para prototipado y gestión de volúmenes moderados de datos (hasta 5 millones de celdas, límite de Sheets).

5.1. Arquitectura de Conexión (Google Apps Script)

Conectar Vue.js directamente a la API v4 de Google Sheets desde el cliente expone las claves API y enfrenta problemas de CORS (Cross-Origin Resource Sharing). La solución arquitectónica correcta es utilizar **Google Apps Script** como intermediario (Proxy API).

Flujo de Datos:

1. **Vue.js (Cliente):** Envía una petición fetch (POST) con el JSON del cliente al endpoint del script (https://script.google.com/macros/s/.../exec).
2. **Google Apps Script (Servidor):**
   - Recibe el doPost(e).
   - Parsea el JSON.
   - Utiliza la clase SpreadsheetApp para acceder a la hoja activa.
   - Ejecuta sheet.appendRow([id, nombre, dni,...]) para insertar los datos.
   - Devuelve un JSON de respuesta { "result": "success", "id": 123 }.
3. **Vue.js (Cliente):** Recibe la confirmación y muestra el modal de éxito.

Este método abstrae la complejidad de OAuth2 y permite una gestión sencilla de los datos. El administrador simplemente abre el Google Sheet y ve las filas aparecer en tiempo real, una interfaz familiar y potente (filtros, ordenación nativa de Excel/Sheets).

5.2. Diseño de la Base de Datos (Esquema de Hojas)

Para mantener la integridad relacional (Cliente 1:N Vehículos) en una hoja de cálculo plana, se proponen dos estrategias:

Estrategia A (Desnormalizada - Más sencilla):

Una sola hoja "Reclamaciones". Si un cliente tiene 2 coches, se crean 2 filas repitiendo los datos del cliente.

- Columnas: ID_Registro, Fecha, DNI, Nombre, Email, Teléfono, Matrícula, Marca, Modelo, Estado.

Estrategia B (Relacional - Más robusta):

Dos hojas: "Clientes" y "Vehiculos".

- **Hoja Clientes:** ID_Cliente (PK), DNI, Nombre, Contacto.
- **Hoja Vehiculos:** ID_Vehiculo, ID_Cliente (FK), Matrícula, Marca, Modelo.
- *Nota:* Implementar esto con Apps Script requiere lógica de búsqueda (vlookup programático) para unir los datos en el Dashboard del administrador.

Dada la naturaleza educativa del proyecto, la **Estrategia A** es preferible por simplicidad, pero la **Estrategia B** demuestra un mayor dominio de la arquitectura de datos.

5.3. Consideraciones de Seguridad

Es fundamental advertir que, aunque funcional, esta arquitectura no es adecuada para datos médicos o financieros de alto riesgo en producción masiva debido a las limitaciones de seguridad de Apps Script (los scripts publicados como "run as me" y accesibles por "anyone" pueden ser invocados por cualquiera que tenga la URL). Para el alcance del proyecto, es aceptable, pero en un informe profesional se debe incluir esta disyuntiva: "Solución válida para prototipo/MVP; migración a Firebase/Supabase recomendada para producción".

---

6. Experiencia de Usuario (UX) y Accesibilidad (A11y)

El perfil demográfico de los afectados por el cártel es transversal: abarca desde jóvenes que compraron su primer coche en 2013 hasta personas mayores que adquirieron vehículos en 2006. Esto hace que la **Accesibilidad Web** sea un requisito no funcional crítico. La aplicación no puede excluir a usuarios con baja competencia digital o discapacidades visuales/motoras.

6.1. Cumplimiento de WCAG 2.1

Basándonos en las mejores prácticas de desarrollo web inclusivo y en las referencias de la OCU [13], la aplicación debe implementar:

- **Semántica HTML5:** Uso estricto de <main>, <section>, <header>, <footer>.
- **Formularios Accesibles:**
  - Cada <input> debe tener un <label> explícitamente asociado mediante el atributo for.
  - Los mensajes de error no deben depender únicamente del color (ej. texto rojo). Deben ir acompañados de iconos y texto descriptivo, vinculados al campo con aria-describedby y aria-invalid.
- **Navegación por Teclado:** El flujo completo de registro debe poder realizarse utilizando únicamente las teclas Tab, Space y Enter. Los elementos interactivos personalizados (como selectores de marca hechos con divs) deben gestionar el foco (tabindex="0") y los eventos de teclado.
- **Contraste de Color:** Asegurar un ratio de contraste mínimo de 4.5:1 para texto normal y 3:1 para texto grande o elementos gráficos (botones). Esto es vital para usuarios con visión reducida o pantallas con mucho reflejo.

6.2. Diseño de Interacción (IxD) para Procesos Legales

El proceso de reclamación genera ansiedad en el usuario. La interfaz debe mitigar esto mediante:

1. **Feedback Constante:** Indicadores de progreso ("Paso 1 de 3: Datos Personales").
2. **Lenguaje Claro:** Evitar jerga legal ("Litisconsorcio", "Solidaridad impropia") en los formularios. Usar términos llanos ("Dueño del coche", "Marca del vehículo").
3. **Prevención de Errores:** Validaciones en tiempo real (mientras se escribe) en lugar de al

enviar el formulario.
4. **Confirmación de Envío:** Una pantalla final clara con un número de referencia y, idealmente, la opción de descargar un PDF resumen de la solicitud (generado en el cliente con librerías como jspdf).

---

7. Plan de Implementación y Despliegue

Para materializar este análisis en un producto funcional, se propone la siguiente hoja de ruta técnica:

Fase 1: Configuración del Entorno (Día 1-2)

- Inicialización del proyecto con npm create vite@latest.
- Instalación de dependencias: vue-router, sass (opcional para estilos).
- Configuración de Git y repositorio GitHub.

Fase 2: Desarrollo de Componentes Base (Día 3-5)

- Creación de BaseInput.vue, BaseSelect.vue, BaseButton.vue.
- Implementación de lógica de accesibilidad en estos componentes base.

Fase 3: Lógica de Negocio y Vistas (Día 6-10)

- Desarrollo de useDniValidation.js y useMatriculaValidation.js.
- Construcción de RegistroCliente.vue y RegistroVehiculo.vue integrando las reglas de validación de fechas del cártel.
- Implementación del sistema de "Store" simple para mantener los datos entre pasos.

Fase 4: Integración Backend (Día 11-13)

- Creación del Google Sheet y el Apps Script (doPost).
- Pruebas de conexión con Postman/Insomnia.
- Integración de llamadas fetch en la aplicación Vue.

Fase 5: Panel de Administración y Refinado (Día 14-15)

- Desarrollo de AdminPanel.vue con tabla de visualización.
- Implementación de filtros por estado (Pendiente/Aceptado).
- Auditoría de accesibilidad con herramientas como *Lighthouse* o *Axe DevTools*.

---

8. Conclusiones y Recomendaciones Finales

El proyecto "Aplicación de Gestión Cártel Coches" representa un desafío técnico de alto nivel

que trasciende la mera programación. Exige del desarrollador una comprensión profunda del contexto legal para implementar reglas de negocio correctas.

La arquitectura propuesta, basada en **Vue.js 3** y **Google Sheets**, ofrece un equilibrio óptimo entre modernidad, rendimiento y coste de desarrollo. Sin embargo, el éxito del proyecto dependerá de la rigurosidad en la implementación de las validaciones de datos. Una aplicación que permita registrar un vehículo "BMW" comprado en 2012 (fuera de plazo) o con una matrícula inválida, fallará en su propósito fundamental de servir como herramienta de pre-cualificación jurídica.

Recomendaciones Clave:

1. **Priorizar la Matriz de Fechas:** Invertir tiempo en crear un archivo de configuración robusto con las fechas exactas de infracción por marca.
2. **Seguridad en el Lado del Cliente:** Implementar validaciones redundantes (frontend y backend script) para asegurar la calidad de los datos.
3. **Enfoque Educativo:** Aprovechar la interfaz para educar al usuario sobre qué documentos necesita (Factura, Modelo 576), convirtiendo la app en una herramienta de asesoramiento además de gestión.

Este informe valida la viabilidad técnica del proyecto y proporciona la base documental necesaria para su ejecución exitosa, alineando los objetivos pedagógicos con una necesidad real del mercado legal español.

Referencias Integradas y Fuentes

- **Especificaciones del Proyecto:**.[1]
- **Contexto Legal y Sentencias:**.[2]
- **Detalles del Cártel (Marcas y Fechas):**.[2]
- **Plazos de Prescripción (2026):**.[4]
- **Tecnología Vue.js 3:**.[18]
- **Integración Google Sheets:**.[20]
- **Validaciones (DNI/Matrícula):**.[24]
- **Documentación Requerida:**.[8]

Works cited

1. Projecte Final Vue.pdf
2. RESOLUCIÓN (Expte. S/0482/13 Fabricantes de automóviles) SALA ..., accessed February 10, 2026, https://www.cnmc.es/sites/default/files/685749_1.pdf
3. Cronología del Cártel de Coches. Cómo hemos llegado a esta situación, accessed February 10, 2026, https://peritojudicial.com/informe-pericial-cartel-coches/cronologia-cartel-coche

s/
4. Plazo de prescripción en el cártel de coches: hasta junio de 2026 puedes reclamar, accessed February 10, 2026, https://pinerorobledilloabogados.es/cartel-coches-plazo-prescripcion-reclamaciones/
5. El TJUE reabre la vía de reclamaciones por el “cártel de coches”. Plazo hasta 2026., accessed February 10, 2026, https://compliancepersonal.com/el-tjue-reabre-la-via-de-reclamaciones-por-el-cartel-de-coches-plazo-hasta-2026/
6. Qué es el cártel de coches y qué marcas están implicadas - Elcano Abogados, accessed February 10, 2026, https://elcanoabogados.com/blog/que-es-el-cartel-de-coches-y-que-marcas-estan-implicadas/page/17/?et_blog
7. Sentencias y Actualidad en el Cártel de Coches - Perito Judicial GROUP, accessed February 10, 2026, https://peritojudicial.com/sentencias-actualidad-cartel-coches/
8. Modelo 576 - Agencia Tributaria, accessed February 10, 2026, https://sede.agenciatributaria.gob.es/Sede/vehiculos-embarcaciones/primera-matriculacion-medios-transporte/modelo-576.html
9. Reclamacion-previa-e-interrupcion-de-la-prescripcion-Cartel-de-coches.docx - Raso & Asociados, accessed February 10, 2026, https://www.rasoyasociados.com/wp-content/uploads/2022/04/Reclamacion-previa-e-interrupcion-de-la-prescripcion-Cartel-de-coches.docx
10. Cártel de Coches II: ¿Qué fechas y qué marcas están afectadas? - Noemi Melio Abogados, accessed February 10, 2026, https://noemimelioabogados.es/cartel-de-coches-ii-que-fechas-y-que-marcas-estan-afectadas/
11. reclamaciones de personas afectadas por el cártel de coches., accessed February 10, 2026, https://www.sevilla.org/servicios/consumo/omic/archivos/cartel-de-automoviles.pdf
12. ¿Cuál es el periodo para reclamar a Marcas sancionadas por Cártel de Coches?, accessed February 10, 2026, https://consultorestecnicos.es/cartel-de-coches-cual-es-el-periodo-por-el-que-se-puede-reclamar-a-cada-una-de-las-marcas-sancionadas/
13. Cartel de Vehículos - OCU, accessed February 10, 2026, https://carteldecoches.ocu.org/
14. Marcas Afectadas por el Cártel de Coches | AACC, accessed February 10, 2026, https://afectadoscartelcoches.es/marcas-afectadas-cartel-coches/
15. Las 13 sentencias del Tribunal Supremo que confirman las multas de la CNMC al cártel de fabricantes de coches | ASJ Jurídico, accessed February 10, 2026, https://www.asjjuridico.com/las-13-sentencias-del-tribunal-supremo-que-confirman-las-multas-de-la-cnmc-al-cartel-de-fabricantes-de-coches/
16. Las 13 sentencias del Tribunal Supremo que confirman las multas de la CNMC al cártel de fabricantes de coches, accessed February 10, 2026,

https://blog.cnmc.es/2022/02/03/las-13-sentencias-del-tribunal-supremo-que-confirman-las-multas-de-la-cnmc-al-cartel-de-fabricantes-de-coches/
17. Nuevo plazo prescripción | AACC - Agrupación de Afectados por el Cártel de Coches, accessed February 10, 2026, https://afectadoscartelcoches.es/nuevo-plazo-prescripcion/
18.
19. Vue 3.2 - Using Composition API with Script Setup - This Dot Labs, accessed February 10, 2026, https://www.thisdot.co/blog/vue-3-2-using-composition-api-with-script-setup
20. Form with Repeatable Group using Vue JS and Google Sheets | Part-II #vuejs #googlesheets - YouTube, accessed February 10, 2026, https://www.youtube.com/watch?v=QbFfONsLK8Q
21. Vue Frontend + GOOGLE SHEETS Backend + How To Deploy For Free On Netlify, accessed February 10, 2026, https://www.youtube.com/watch?v=65Mu9F67758
22. Leer archivo Google Sheets desde JS (Vuejs) | by Alejandro Oñate | Medium, accessed February 10, 2026, https://codeinspiracion.medium.com/leer-archivo-google-sheets-desde-js-vue-ecd676688e2f
23. Build a CRUD API using the Google Sheets API - LogRocket Blog, accessed February 10, 2026, https://blog.logrocket.com/build-crud-api-using-google-sheets-api/
24. Número de identificación fiscal - Wikipedia, la enciclopedia libre, accessed February 10, 2026, https://es.wikipedia.org/wiki/N%C3%BAmero_de_identificaci%C3%B3n_fiscal
25. Matrículas automovilísticas de España - Wikipedia, la enciclopedia ..., accessed February 10, 2026, https://es.wikipedia.org/wiki/Matr%C3%ADculas_automovil%C3%ADsticas_de_Espa%C3%B1a
26. Función para validar DNI (NIF), CIF y NIE con JavaScript - Jorge Rosa | Blog, accessed February 10, 2026, https://jorgerosa.dev/posts/funcion-para-validar-nif-nie-cif-con-javascript/
27. Documentos necesarios para Reclamar en el Cártel de Coches - AACC, accessed February 10, 2026, https://afectadoscartelcoches.es/documentacion-cartel-coches/
28. Documentacion necesaria para reclamar el CÁRTEL DE COCHES [2023] AACC - YouTube, accessed February 10, 2026, https://www.youtube.com/watch?v=7u0B9YUM2vI