

ICAI – UNIVERSIDAD PONTIFICIA COMILLAS

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Plataforma digital integral para el monoplaza eléctrico de ISC Racing Team: arquitectura web de datos e inteligencia de telemetría embarcada con Edge Computing

Autora: Gabriela de Dorremocha

Director: Vidal Domínguez, Mario

Co-Director: Sanz Bobi, Miguel Ángel

Madrid

Junio de 2026

Declaración de Originalidad

Yo, Gabriela de Dorremochea, estudiante del Grado en Ingeniería en Tecnologías de Telecomunicación de la ICAI – Universidad Pontificia Comillas, declaro bajo mi responsabilidad que el presente Trabajo Fin de Grado titulado:

Plataforma digital integral para el monoplaça eléctrico de ISC Racing Team:
arquitectura web de datos e inteligencia de telemetría embarcada con Edge Computing

es original, fruto de mi trabajo personal, y que no ha sido presentado con anterioridad para la obtención de ningún título o calificación. Todas las fuentes, ideas y aportaciones de terceros utilizadas han sido debidamente citadas y referenciadas conforme a la normativa académica vigente.

Uso de herramientas de Inteligencia Artificial

Durante la realización de este trabajo se han empleado herramientas de Inteligencia Artificial únicamente como apoyo en tareas auxiliares, en concreto: la organización y planificación de las tareas del proyecto, la búsqueda de ideas y enfoques, la revisión de los documentos para comprobar el cumplimiento de los requisitos de la normativa, y la ayuda en la corrección de errores de programación. La redacción, el contenido técnico, las decisiones de diseño, los resultados experimentales y su interpretación son obra y responsabilidad de la autora.

La autora

Gabriela de Dorremochea

Fecha: 23/06/2026



Vº Bº del Director

Vidal Domínguez, Mario

Fecha: 23/06/2026



Vº Bº del Codirector

Sanz Bobi, Miguel Ángel

Fecha: _____

Agradecimientos

A mis directores, Mario Vidal Domínguez y Miguel Ángel Sanz Bobi, por su orientación a lo largo de este trabajo, por el criterio con el que han revisado cada etapa y por la exigencia que ha hecho mejor esta memoria.

A ISC Racing Team, por darme la oportunidad de aprender resolviendo problemas reales y de aportar algo útil al equipo. A mis compañeros y compañeras de las divisiones de Coche y de Moto, por su trabajo y por confiar en la plataforma que aquí se describe.

A los profesores del Grado en Ingeniería en Tecnologías de Telecomunicación y del programa de *Business Analytics* de ICAI, cuya formación es la base de este proyecto.

Y, sobre todo, a mi familia y a mis amigos, por su apoyo constante y su paciencia a lo largo de estos años. Gracias.

Resumen

PLATAFORMA DIGITAL INTEGRAL PARA EL MONOPLAZA ELÉCTRICO DE ISC RACING TEAM: ARQUITECTURA WEB DE DATOS E INTELIGENCIA DE TELEMETRÍA EMBARCADA CON EDGE COMPUTING

Autora: De Dorremocha, Gabriela

Director: Vidal Domínguez, Mario

Codirector: Sanz Bobi, Miguel Ángel

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Introducción

El trabajo responde a una doble necesidad real de ISC Racing Team, equipo de Formula Student y MotoStudent con un monoplaza eléctrico: dominar el dato del equipo de extremo a extremo, desde la infraestructura que lo capta y lo presenta hasta el motor que lo explota. Para ello desarrolla dos subsistemas complementarios. El primero es una **plataforma web** profesional, orientada a la captación de talento y de patrocinio B2B, que erradica la duplicación de código heredada de mantener por separado las divisiones de Coche y Moto mediante un patrón de componente compartido con wrappers finos y una capa de datos editable de forma autónoma. El segundo es un **sistema de inteligencia de telemetría** basado en aprendizaje automático y Edge Computing: el análisis de telemetría es una disciplina madura en la alta competición, pero en Formula Student sigue siendo mayoritariamente descriptivo, manual y casi nunca embarcado, y la inferencia en el extremo exige un requisito a menudo subestimado (la causalidad de las características) sin el cual una métrica de laboratorio carece de validez en el vehículo. El objeto de este subsistema es construir un pipeline de telemetría reproducible y libre de fuga de información, una arquitectura dual de modelos Box/Edge y una validación honesta de su capacidad de generalización.

Metodología

El subsistema web se desarrolló de forma incremental e iterativa (análisis, refactorización arquitectónica, reorganización por dominios, calidad y documentación), construido sobre React 19, Vite y SCSS con un sistema de tokens de marca, y desplegado de forma continua en Vercel. La fase de modelado del subsistema de telemetría se estructuró como un pipeline de cinco fases, adquisición, limpieza por grupos, ingeniería de características causales, validación cruzada y test aislado, sobre telemetría pública de Fórmula 1 (`fastf1`) empleada como banco de pruebas trasladable al monoplaza eléctrico. El problema se formaliza como una clasificación multiclase: predecir la marcha engranada a partir de la telemetría instantánea. Se diseñó una arquitectura dual: un MLP de alta capacidad para el análisis en el Box y un árbol de decisión ligero para la inferencia en tiempo real en el Edge (Raspberry Pi). La validación es una validación cruzada por circuitos (`leave-one-circuit-out`) que entrena en Monza, Spa y Silverstone y reserva Mónaco como test aislado. Todas las características son causales, verificadas mediante la invariancia de prefijo, y la reproducibilidad se garantiza fijando semillas y serializando los parámetros de inferencia congelados.

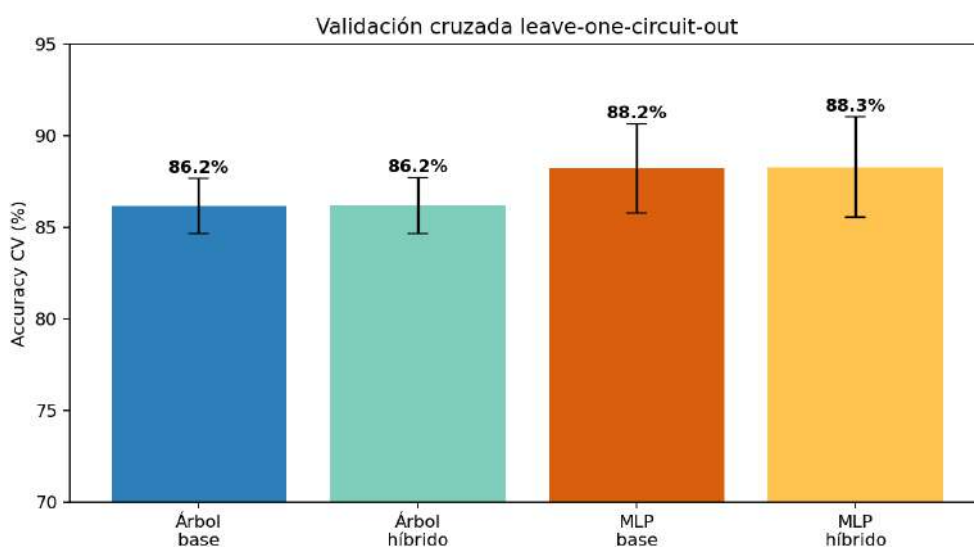


Figura 1. Esquema de validación cruzada por circuitos (`leave-one-circuit-out`): entrenamiento en Monza, Spa y Silverstone y test aislado en Mónaco. El MLP del Box supera al árbol del Edge de forma consistente pero modesta.

Resultados

El subsistema web culmina en una plataforma de veinticuatro rutas para ambas disciplinas, con una presentación cuidada, un sistema de marca uniforme y un diseño responsive validado (figura 2); la unificación erradica la duplicación de componentes y reduce a la mitad el coste de mantenimiento por construcción.



Figura 2. Portada (Home) de la división de Formula Student en la plataforma web del ISC Racing Team, una de las veinticuatro páginas del sitio.

En el subsistema de telemetría, la validación cruzada honesta sitúa al MLP en 88,24 % de accuracy y al árbol en 86,18 %. En el test aislado de Mónaco (un genuino domain shift) el MLP retiene un 78,05 % y el árbol un 75,11 %, con los errores concentrados en marchas adyacentes. Un experimento controlado cuantifica en 7,52 puntos el sesgo optimista que introduciría una partición barajada, lo que valida la elección metodológica. El PCA reconstruye sin supervisión los cuatro modos físicos del vehículo, y el cociente régimen/velocidad domina la importancia de características, evidencia de que el modelo aprende la física de la transmisión. El árbol del Edge ocupa 31,7 kB e infiere en 1,90 ms por muestra, frente a los 328,1 kB del MLP, cumpliendo con holgura el presupuesto de 10 ms del lazo de control a 100 Hz.

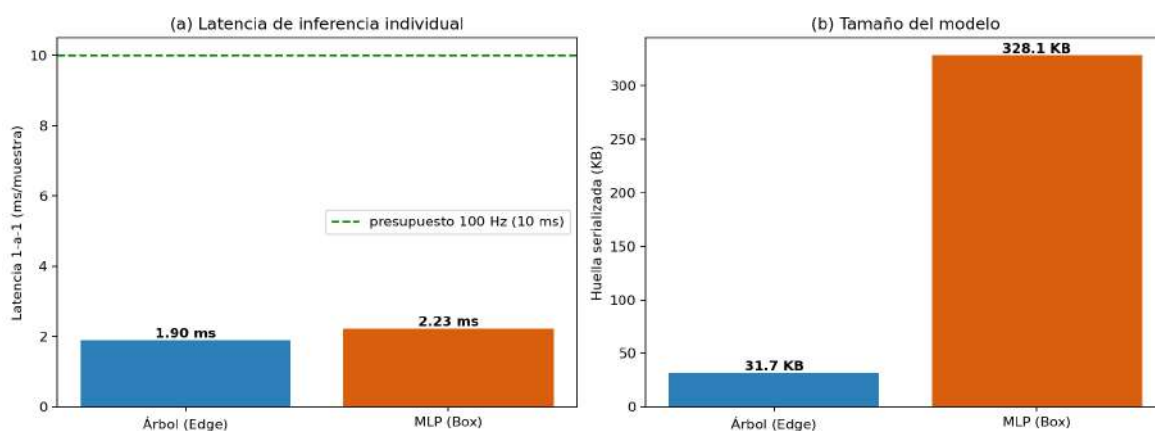


Figura 3. Viabilidad del Edge: (a) latencia de inferencia individual frente al presupuesto de 10 ms del lazo a 100 Hz; (b) huella de memoria serializada. El árbol cumple ambos requisitos con amplio margen.

Conclusiones

El subsistema web devuelve al equipo la autonomía para mantener una plataforma profesional, limpia y coherente, demostrando que reformular el problema, misma estructura, distinto contenido, elimina la duplicación de raíz. En el subsistema de telemetría, la arquitectura dual Box/Edge queda justificada por evidencia cuantitativa de precisión, latencia y huella de memoria: el rigor metodológico se demuestra, no se afirma, y el sistema generaliza de forma honesta a un circuito nunca visto porque captura la física del tren motriz y no las particularidades de un trazado; la penalización de unos 3 puntos por embarcar el árbol en lugar del MLP es un precio razonable a cambio de autonomía, determinismo y frugalidad, y la metodología es directamente trasladable al monoplaza eléctrico. Ambos subsistemas comparten una misma estrategia de dominio del dato, separación entre dato y lógica, frugalidad y autonomía, y convergen, como línea futura, en un panel de visualización de la telemetría embarcado en la propia plataforma, todo ello con un coste material prácticamente nulo.

Palabras clave

plataforma web; React; arquitectura de componentes; telemetría; machine learning; edge computing; powertrain eléctrico; Formula Student; validación cruzada por circuitos.

Referencias

Las fuentes que sustentan este resumen se citan a lo largo del cuerpo de la memoria y se reconocen, con formato IEEE, en el capítulo de Bibliografía. Las referencias principales son la librería de telemetría `fastf1`, la biblioteca `scikit-learn` para el modelado y la literatura de Edge Computing y validación honesta en aprendizaje automático [1], [2].

Abstract

END-TO-END DIGITAL PLATFORM FOR ISC RACING TEAM'S ELECTRIC RACE CAR: WEB DATA ARCHITECTURE AND EMBEDDED TELEMETRY INTELLIGENCE WITH EDGE COMPUTING

Author: De Dorremocha, Gabriela

Supervisor: Vidal Domínguez, Mario

Co-Supervisor: Sanz Bobi, Miguel Ángel

Collaborating Entity: ICAI – Universidad Pontificia Comillas

PROJECT SUMMARY

Introduction

This work responds to a twofold real need of ISC Racing Team, a Formula Student and MotoStudent team with an electric race car: to master the team's data end to end, from the infrastructure that captures and presents it to the engine that exploits it. To this end it develops two complementary subsystems. The first is a professional **web platform**, aimed at recruiting talent and attracting B2B sponsors, which eradicates the code duplication inherited from maintaining the Car and Motorcycle divisions separately, by means of a shared-component pattern with thin wrappers and an autonomously editable data layer. The second is a **telemetry intelligence system** based on machine learning and Edge Computing: telemetry analysis is a mature discipline in top-tier motorsport, but in Formula Student it remains largely descriptive, manual and rarely embedded, and edge inference imposes an often underestimated requirement (feature causality) without which a laboratory metric is invalid on the car. The aim of this subsystem is to build a reproducible, leakage-free telemetry pipeline, a dual Box/Edge model architecture and an honest validation of its generalization capability.

Methodology

The web subsystem was developed incrementally and iteratively (analysis, architectural refactoring, domain-based reorganization, quality and documentation), built on React 19, Vite and SCSS with a brand-token system, and continuously deployed on Vercel. The modelling phase of the telemetry subsystem was structured as a five-stage pipeline, acquisition, group-wise cleaning, causal feature engineering, cross-validation and isolated test, over public Formula 1 telemetry (`fastf1`) used as a testbed transferable to the electric car. The problem is cast as a multiclass classification: predicting the engaged gear from instantaneous telemetry. A dual architecture was designed: a high-capacity MLP for garage (Box) analysis and a lightweight decision tree for real-time inference at the Edge (Raspberry Pi). Validation is a leave-one-circuit-out cross-validation trained on Monza, Spa and Silverstone, with Monaco held out as an isolated test. All features are causal, verified through prefix invariance, and reproducibility is ensured by fixing seeds and serializing the frozen inference parameters.

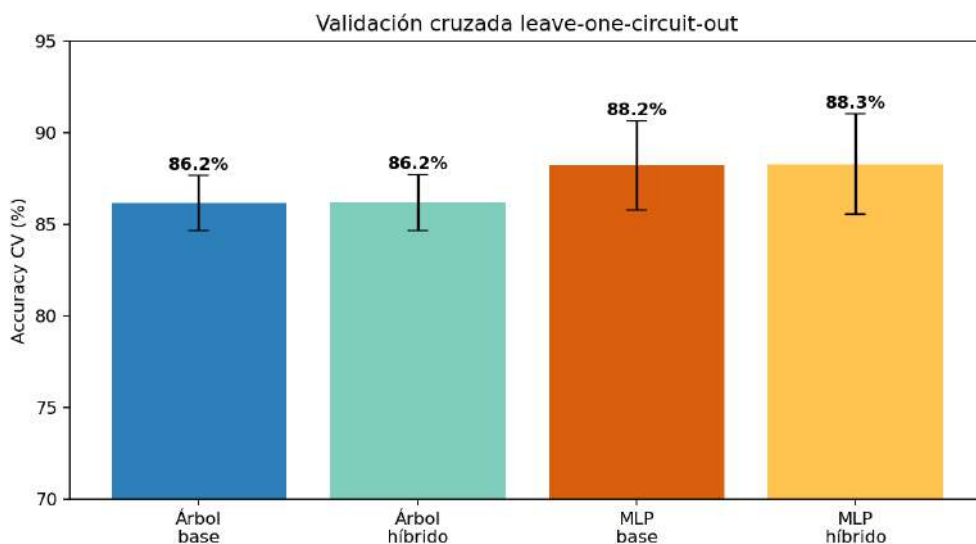


Figure 4. Leave-one-circuit-out cross-validation scheme: training on Monza, Spa and Silverstone, with Monaco held out as an isolated test. The Box MLP consistently but modestly outperforms the Edge tree.

Results

The web subsystem culminates in a twenty-four-route platform for both disciplines, with a polished and consistent finish, a uniform brand system and a validated responsive design (figura 5); unification eradicates component duplication and halves maintenance cost by construction.



Figure 5. Home page of the Formula Student division on ISC Racing Team’s web platform, one of the twenty-four pages of the site.

In the telemetry subsystem, honest cross-validation places the MLP at 88,24 % accuracy and the tree at 86,18 %. On the isolated Monaco test (a genuine domain shift) the MLP retains 78,05 % and the tree 75,11 %, with errors concentrated on adjacent gears. A controlled experiment quantifies at 7,52 points the optimistic bias a shuffled split would introduce, which validates the methodological choice. PCA reconstructs the four physical operating modes of the vehicle without supervision, and the engine-speed-to-velocity ratio dominates feature importance, evidence that the model learns the physics of the transmission. The Edge tree occupies 31,7 kB and infers in 1,90 ms per sample, against the 328,1 kB of the MLP, comfortably meeting the 10 ms budget of a 100 Hz control loop.

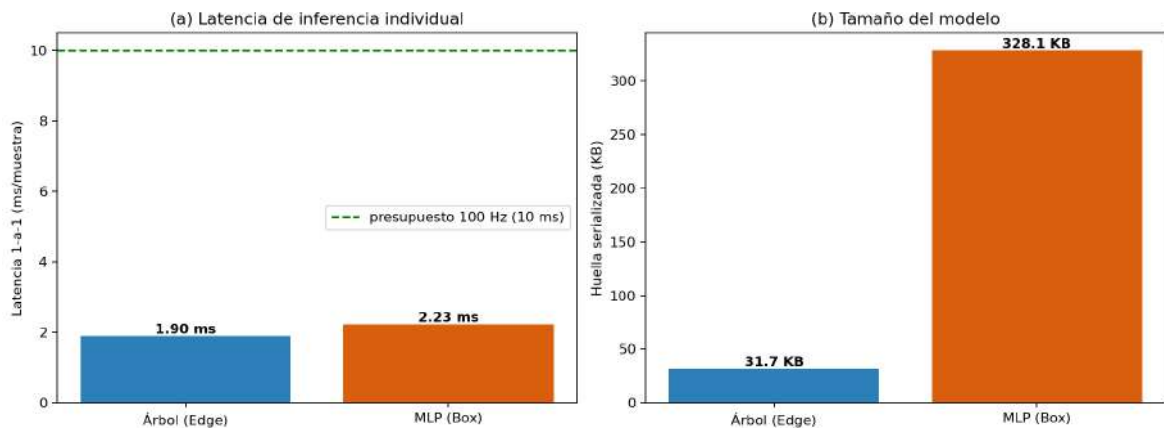


Figure 6. Edge viability: (a) per-sample inference latency against the 10 ms budget of a 100 Hz loop; (b) serialized memory footprint. The tree meets both requirements with ample margin.

Conclusions

The web subsystem restores the team’s autonomy to maintain a professional, clean and coherent platform, showing that reframing the problem, same structure, different content, removes duplication at the root. In the telemetry subsystem, the dual Box/Edge architecture is justified by quantitative evidence on accuracy, latency and memory footprint: methodological rigour is demonstrated, not asserted, and the system generalizes honestly to an unseen circuit because it captures the physics of the powertrain rather than the idiosyncrasies of a track; the roughly 3-point penalty of deploying the tree instead of the MLP is a reasonable price for autonomy, determinism and frugality, and the methodology transfers directly to the electric car. Both subsystems share a single data-mastery strategy, separation of data and logic, frugality and autonomy, and converge, as future work, on a telemetry visualization panel embedded in the platform itself, all at a near-zero material cost.

Keywords

web platform; React; component architecture; telemetry; machine learning; edge computing; electric powertrain; Formula Student; circuit-based cross-validation.

References

The sources supporting this abstract are cited throughout the body of the report and listed, in IEEE format, in the Bibliography chapter. The main references are the `fastf1` telemetry library, the `scikit-learn` modelling toolkit and the literature on Edge Computing and honest validation in machine learning [1], [2].

Índice general

Declaración de Originalidad	III
Agradecimientos	V
Resumen	VII
Abstract	XI
Lista de Acrónimos	XXV
1. Introducción	1
1.1. Motivación y contexto	1
1.2. Una plataforma de datos de extremo a extremo: el hilo conductor	2
1.3. Objeto del proyecto	3
1.4. Estructura de la memoria	4
2. Descripción de las Tecnologías	7
2.1. El stack de la plataforma web	7
2.1.1. Aplicaciones de página única (SPA)	7
2.1.2. React 19 y el paradigma de componentes	8
2.1.3. Vite, React Router y SCSS con sistema de tokens	8
2.1.4. El contenido estático como gestor de contenidos ligero	9
2.2. Adquisición de telemetría	9
2.2.1. El bus CAN como origen del dato	9
2.2.2. La telemetría de Fórmula 1 como banco de pruebas: <code>fastf1</code>	10
2.3. Aprendizaje automático: fundamentos y stack	10
2.3.1. Formalización del problema de clasificación	11
2.3.2. Árbol de decisión	11
2.3.3. Perceptrón multicapa (MLP)	11
2.3.4. Técnicas no supervisadas: PCA y K-Means	12
2.4. Edge Computing: la Raspberry Pi del volante	12

3. Estado de la Cuestión	15
3.1. Plataformas web de equipos de competición	15
3.2. Análisis de telemetría en motorsport	16
3.3. Inferencia embarcada en tiempo real (Edge AI)	16
3.4. Honestidad metodológica: fuga de información y causalidad	17
3.5. Aportación diferencial de este trabajo	17
4. Definición del Trabajo	19
4.1. Justificación	19
4.2. Objetivos	20
4.3. Metodología	21
4.4. Planificación temporal	22
4.5. Estimación económica	23
5. Subsistema I: la plataforma web de datos	25
5.1. Análisis	25
5.2. Diseño	27
5.2.1. El patrón componente compartido + wrappers finos	27
5.2.2. La carpeta data/ como gestor de contenidos estático	28
5.2.3. Normalización de rutas, organización por dominios y sistema de marca	28
5.3. Implementación	29
5.3.1. Navegación y pie de página	30
5.3.2. Portada (Home)	31
5.3.3. Garaje y ficha de vehículo	32
5.3.4. Historia y línea temporal	33
5.3.5. Equipo y organigrama	34
5.3.6. Patrocinadores y captación de patrocinio	35
5.3.7. Competiciones y mapa de resultados	36
5.3.8. Actualidad y boletines	36
5.3.9. Únete y contacto	37
5.3.10. Diseño responsive	38
5.4. Gestión de contenido para perfiles no técnicos	39
6. Subsistema II: inteligencia de telemetría (Box/Edge)	41
6.1. Análisis	41
6.2. Diseño	42
6.2.1. Arquitectura dual Box / Edge	42
6.2.2. Ingeniería de características causales	42
6.3. Implementación	43
6.3.1. Adquisición y limpieza	43

6.3.2.	Validación cruzada por circuitos	44
6.3.3.	Modelos: MLP (Box) y Árbol de Decisión (Edge)	45
7.	Análisis de Resultados	49
7.1.	Resultados de la plataforma web	49
7.1.1.	Métricas de código y reducción de la duplicación	49
7.1.2.	Caso de estudio: el fallo de color del CTA	50
7.1.3.	Grado de cumplimiento de los objetivos del subsistema web	50
7.2.	Resultados del subsistema de telemetría	51
7.2.1.	Conjuntos de entrenamiento y test	51
7.2.2.	Generalización: del entrenamiento al test aislado en Mónaco	52
7.2.3.	Interpretación física del PCA	54
7.2.4.	Rendimiento de modelos: Árbol de Decisión frente a MLP	55
7.2.5.	Viabilidad de la arquitectura Edge: latencia y huella	57
7.2.6.	Valoración económica: el valor de la herramienta para el equipo	58
8.	Conclusiones y Trabajos Futuros	61
8.1.	Conclusiones	61
8.1.1.	Sobre la plataforma web	61
8.1.2.	Sobre el subsistema de inteligencia de telemetría	61
8.1.3.	Conclusión integradora: el dominio del dato de extremo a extremo	62
8.2.	Limitaciones del trabajo	63
8.3.	Trabajos futuros	64
8.3.1.	Líneas del subsistema web	64
8.3.2.	Líneas del subsistema de telemetría	64
8.3.3.	Línea integradora: el panel de telemetría en la plataforma	65
	Bibliografía	67
A.	Alineación del proyecto con los Objetivos de Desarrollo Sostenible	69
B.	Material complementario	71
B.1.	Disponibilidad del código fuente	71
B.2.	Fragmentos de código relevantes	71
B.3.	Tablas extensas	73
B.4.	Fragmentos de código del subsistema web	73

Índice de figuras

1.	Esquema de validación cruzada por circuitos (leave-one-circuit-out): entrenamiento en Monza, Spa y Silverstone y test aislado en Mónaco. El MLP del Box supera al árbol del Edge de forma consistente pero modesta.	VIII
2.	Portada (Home) de la división de Formula Student en la plataforma web del ISC Racing Team, una de las veinticuatro páginas del sitio.	IX
3.	Viabilidad del Edge: (a) latencia de inferencia individual frente al presupuesto de 10 ms del lazo a 100 Hz; (b) huella de memoria serializada. El árbol cumple ambos requisitos con amplio margen.	IX
4.	Leave-one-circuit-out cross-validation scheme: training on Monza, Spa and Silverstone, with Monaco held out as an isolated test. The Box MLP consistently but modestly outperforms the Edge tree.	XII
5.	Home page of the Formula Student division on ISC Racing Team's web platform, one of the twenty-four pages of the site.	XIII
6.	Edge viability: (a) per-sample inference latency against the 10 ms budget of a 100 Hz loop; (b) serialized memory footprint. The tree meets both requirements with ample margin.	XIII
1.1.	Visión de conjunto de la plataforma de datos de extremo a extremo de ISC Racing Team. El subsistema de telemetría (motor de inteligencia) transforma el dato crudo del bus CAN en predicciones mediante una arquitectura dual Box/Edge; el subsistema web (infraestructura de datos y comunicación) capta talento y patrocinio y constituye la superficie de presentación y visualización del dato del equipo.	3
4.1.	Diagrama de Gantt de la planificación del proyecto a lo largo del curso 2025–2026. El primer cuatrimestre (azul) se dedicó al desarrollo de la plataforma web; el segundo, al modelado de telemetría (naranja), a la interpretabilidad y viabilidad en el Edge (verde) y a los detalles finales de la web (turquesa), con la redacción de la memoria (morado) solapada. La línea de puntos marca el hito de validación del pipeline honesto.	23

5.1. Página de aterrizaje (Landing Split): bifurcación del visitante entre la división de Coche (Formula Student) y la de Moto (MotoStudent).	26
5.2. Barra de navegación del Coche, con el menú y los colores de la disciplina activa, y pie de página con enlaces y redes sociales.	31
5.3. Portada (Home) del Coche: portada multimedia a pantalla completa, presentación del equipo y contadores numéricos animados.	32
5.4. Garaje: cuadrícula de vehículos, y ficha de detalle de un vehículo con especificaciones, galería e historial de competición.	33
5.5. Historia: línea temporal con la evolución de los vehículos y ampliación de imágenes a pantalla completa.	34
5.6. Organigrama del equipo con filtrado por departamento y enlaces a los perfiles profesionales de cada miembro.	34
5.7. Patrocinadores agrupados por niveles.	35
5.8. Página «Patrocínanos»: propuesta de valor B2B por niveles (tiers) con descarga del dossier de patrocinio.	35
5.9. Competiciones: mapa interactivo de competiciones (Leaflet) con marcadores para eventos asistidos y objetivos futuros, junto al palmarés histórico. . . .	36
5.10. Actualidad: listado de boletines con su portada y visor de PDF integrado. .	37
5.11. Páginas de captación de talento (Únete) y de contacto: formulario de inscripción, formulario de contacto y mapa de ubicación.	38
5.12. Comparativa de la página de aterrizaje en escritorio y en móvil (diseño responsive): el contenido se reordena y las imágenes mantienen sus proporciones.	39
6.1. Aceleración longitudinal causal frente a centrada. La señal causal (azul) es invariante al truncar el futuro de la vuelta en el corte de prefijo, condición necesaria para su despliegue en el Edge; la centrada (rojo) usa información futura y cambia al truncar.	43
6.2. Cuantificación del sesgo por fuga de información. La partición barajada (con fuga) sobreestima la accuracy en 7,52 puntos frente a la partición honesta por circuitos.	45
7.1. Análisis exploratorio del conjunto de entrenamiento: (a) desequilibrio de clases de la marcha; (b) bandas diagonales por marcha en el plano velocidad–RPM, evidencia de que el problema es aprendible; (c) traza de velocidad de una vuelta.	52
7.2. Validación cruzada leave-one-circuit-out. El MLP supera al árbol de forma consistente pero modesta; la variante híbrida no aporta mejora significativa.	53
7.3. Matrices de confusión en el test aislado de Mónaco. Los errores se concentran en marchas adyacentes a la diagonal, el error menos grave.	54

7.4.	Detección de regímenes de conducción: (a) gráfico de sedimentación que identifica cuatro componentes para el 90 % de varianza; (b) método del codo para K-Means; (c) regímenes en el plano de las dos primeras componentes.	54
7.5.	Importancia de las características del árbol del Edge. El cociente rpm_per_speed domina, reconstruyendo la relación física de la caja de cambios.	56
7.6.	Primera división del árbol del Edge. La decisión raíz es un umbral sobre el cociente régimen/velocidad, legible y físicamente interpretable.	56
7.7.	Árbol de decisión completo del Edge (profundidad 5, 9 hojas). Cada nodo muestra el umbral de decisión, el porcentaje de muestras y la clase mayoritaria. El modelo decide casi por completo a partir del cociente régimen/velocidad (rpm_per_speed) y, en menor medida, de la aceleración longitudinal (a_lon).	57
7.8.	Viabilidad del Edge: (a) latencia de inferencia individual frente al presupuesto de 10 ms del lazo a 100 Hz; (b) huella de memoria serializada. El árbol cumple ambos requisitos con amplio margen.	58

Índice de tablas

4.1. Recursos del proyecto y su coste para el equipo.	24
5.1. Mapa de rutas de la plataforma (veinticuatro rutas en total).	26
5.2. Tokens principales del sistema de diseño (<code>_variables.scss</code>).	28
5.3. Extracto de la tabla operativa del manual para Marketing.	39
6.1. Arquitectura dual Box/Edge: dos cerebros para dos tiempos de respuesta. . .	42
6.2. Barrido de profundidad del árbol del Edge (exactitud de validación cruzada <i>leave-one-circuit-out</i>).	46
6.3. Rejilla de arquitectura y regularización del MLP del Box (exactitud de validación cruzada <i>leave-one-circuit-out</i>).	46
6.4. Métricas de entrenamiento y de validación cruzada <i>leave-one-circuit-out</i> de ambos modelos.	47
7.1. Distribución aproximada de líneas de código por área del subsistema web. .	50
7.2. Grado de cumplimiento de los objetivos del subsistema web.	51
7.3. Composición de los conjuntos de entrenamiento y test.	51
7.4. Distribución de la marcha engranada (nGear) por conjunto.	52
B.1. Resultados de la validación cruzada <i>leave-one-circuit-out</i> y del test aislado de Mónaco. Exactitud en % y F1 ponderada.	73
B.2. Métricas de viabilidad del Edge frente al Box.	73

Lista de Acrónimos

API Application Programming Interface, interfaz de programación.

B2B Business to Business, relación comercial entre empresas.

CAN Controller Area Network, bus de comunicaciones del vehículo.

CMS Content Management System, gestor de contenidos.

CSS Cascading Style Sheets, hojas de estilo en cascada.

CSV Comma-Separated Values, formato de datos tabulares en texto plano.

CTA Call To Action, llamada a la acción.

CV Cross-Validation, validación cruzada.

DOM Document Object Model, modelo de objetos del documento.

DRS Drag Reduction System, sistema de reducción de resistencia aerodinámica.

EV Electric Vehicle, vehículo eléctrico.

JSX JavaScript XML, extensión sintáctica de JavaScript usada en React.

ML Machine Learning, aprendizaje automático.

MLP Multi-Layer Perceptron, perceptrón multicapa.

ODS Objetivos de Desarrollo Sostenible.

PCA Principal Component Analysis, análisis de componentes principales.

ReLU Rectified Linear Unit, función de activación del MLP.

RPM Revoluciones por minuto del motor.

SCSS Sassy CSS, extensión del CSS con variables y anidamiento.

SI Sistema Internacional de Unidades.

SPA Single Page Application, aplicación de página única.

TFG Trabajo Fin de Grado.

UI/UX User Interface / User Experience, interfaz y experiencia de usuario.

Capítulo 1

Introducción

1.1. Motivación y contexto

La competición Formula Student, y su homóloga sobre dos ruedas, MotoStudent, constituye uno de los entornos formativos más exigentes de la ingeniería universitaria: equipos íntegramente formados por estudiantes diseñan, fabrican y compiten con un vehículo concebido por ellos mismos, sometido a pruebas dinámicas y estáticas que evalúan tanto el rendimiento en pista como la solidez de las decisiones de ingeniería, el plan de negocio y la capacidad del equipo para captar recursos. En este contexto se enmarca el **ISC Racing Team** de la Universidad Pontificia Comillas (ICAI), que reúne bajo una misma marca dos divisiones técnicamente independientes pero hermanadas: la división de Formula Student, que desarrolla un monoplaza **cien por cien eléctrico**, y la división de MotoStudent, que desarrolla un prototipo de motocicleta eléctrica. La electrificación ha convertido la gestión inteligente de la energía en el factor diferencial de la prueba, y ha hecho del **dato** el activo estratégico del equipo.

Ese activo se manifiesta en dos planos que, a primera vista, parecen inconexos pero que comparten una misma naturaleza. En el plano **externo**, la presencia digital del equipo (su plataforma web) es la herramienta que sostiene las dos funciones que garantizan su supervivencia: la captación de talento de ingeniería, que asegura el relevo generacional, y la captación de patrocinadores, un proceso de relación entre empresas (B2B) en el que una presencia digital poco cuidada puede dificultar la obtención de acuerdos. En el plano **interno**, el monoplaza genera, a través de su bus de comunicaciones Controller Area Network (CAN), un caudal de telemetría de alta frecuencia, velocidad de rueda, régimen de motor, posición del acelerador y del freno, marcha engranada, temperaturas y tensiones, cuyo valor solo se materializa si se transforma en **información accionable**. Uno y otro plano son, en el fondo, el mismo problema de ingeniería del dato: capturarlo, estructurarlo, procesarlo y presentarlo de forma que genere valor para el equipo.

El presente Trabajo Fin de Grado (TFG) responde a esta doble necesidad real del equipo

ISC, surgida del papel de la autora dentro del equipo, primero en el área de sistemas y actualmente como **jefa de digitalización**, y aborda el ciclo de vida del dato del equipo de **extremo a extremo** mediante dos subsistemas complementarios. Desde esa responsabilidad, el trabajo se concibe además como la primera fase de una hoja de ruta de digitalización más ambiciosa para el equipo, cuyas líneas futuras, en particular la construcción de un backend con acceso autenticado para los miembros del equipo, se detallan en el Capítulo 8. El primero es una **plataforma web** profesional y coherente, concebida como la infraestructura de captación, organización y presentación de los datos públicos del equipo, y diseñada para que un perfil no técnico pueda mantenerla de forma autónoma. El segundo es un **sistema de inteligencia de telemetría** basado en aprendizaje automático (Machine Learning, ML) y en el paradigma de Edge Computing, que actúa como el motor de predicción que convierte la telemetría cruda del monoplaza en decisiones, tanto en el análisis reposado del garaje como en la inferencia embarcada en tiempo real sobre el propio vehículo. El trabajo se nutre del doble grado en Ingeniería en Tecnologías de Telecomunicación y Business Analytics, combinando la ingeniería del software y de sistemas con la perspectiva de análisis de datos y de decisión.

1.2. Una plataforma de datos de extremo a extremo: el hilo conductor

Aunque los dos subsistemas emplean tecnologías distintas, desarrollo web frontend en un caso, aprendizaje automático y computación embarcada en el otro, no son dos proyectos yuxtapuestos sino las **dos caras de una única estrategia de datos** para el monoplaza eléctrico de ISC. La figura 1.1 sintetiza esta visión de conjunto: el dato nace en el vehículo (bus CAN y sensores), se procesa y se convierte en inteligencia mediante la arquitectura dual Box/Edge de ML, y se proyecta hacia el exterior, resultados, evolución de los vehículos, rendimiento, a través de la plataforma web, que es a la vez la superficie pública que da soporte a la captación de patrocinio y de talento y la futura superficie de visualización de esa inteligencia.

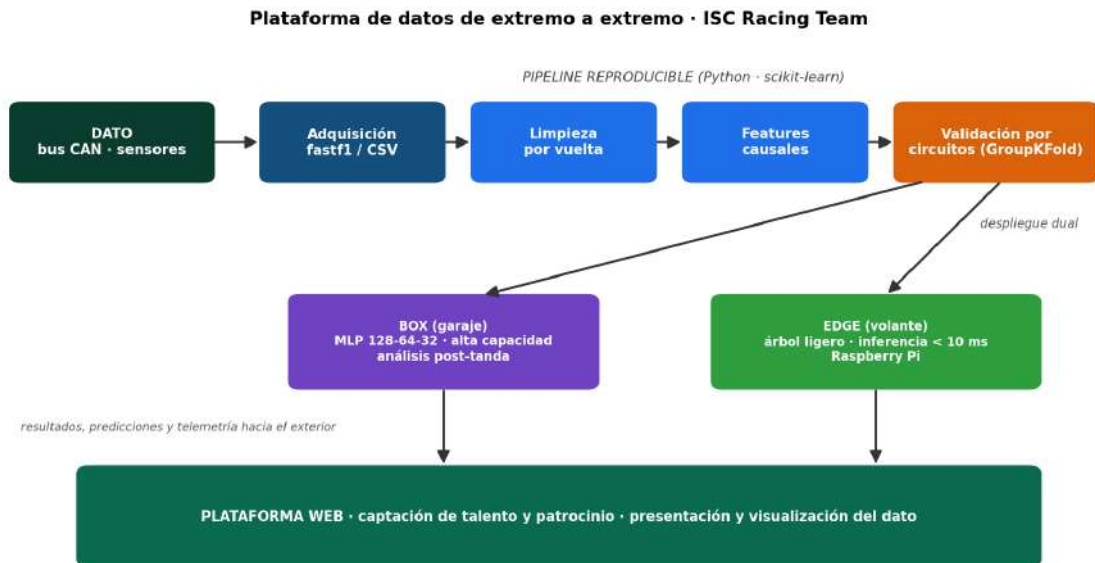


Figura 1.1. Visión de conjunto de la plataforma de datos de extremo a extremo de ISC Racing Team. El subsistema de telemetría (motor de inteligencia) transforma el dato crudo del bus CAN en predicciones mediante una arquitectura dual Box/Edge; el subsistema web (infraestructura de datos y comunicación) capta talento y patrocinio y constituye la superficie de presentación y visualización del dato del equipo.

La articulación entre ambos subsistemas se produce en tres puntos concretos. Primero, en la **filosofía de ingeniería de datos** compartida: tanto la externalización del contenido editable a una capa de datos versionada en la web como la prevención rigurosa de la fuga de información en el pipeline de ML responden al mismo principio, separar el dato de la lógica que lo consume y garantizar su integridad. Segundo, en la **capa de presentación**: la plataforma web es la candidata natural para alojar, en su evolución futura, un panel de visualización de la telemetría procesada y de las predicciones del modelo, cerrando el ciclo entre la inteligencia generada a bordo y su comunicación al equipo y a los patrocinadores. Tercero, en una **misma exigencia de frugalidad y autonomía**: la web se diseña para no depender de un servidor ni de licencias, y el modelo embarcado para inferir sobre hardware de bajo coste sin conexión externa, de modo que el equipo conserve el control íntegro de su dato a un coste material prácticamente nulo. Esta unidad de propósito, el dominio del dato del equipo, desde su captura hasta su explotación, es el hilo conductor que recorre la memoria de principio a fin.

1.3. Objeto del proyecto

El objeto de este trabajo es el **diseño, la implementación y la validación de la plataforma de datos de ISC Racing Team**, descompuesta en dos subsistemas con objetivos propios pero subordinados a la estrategia común descrita.

El **primer subsistema** es la plataforma web oficial del equipo. Su objeto es elevar la calidad visual y la coherencia de la web, orientada a la captación de talento y de patrocinio B2B y, simultáneamente, rediseñar su arquitectura de software para erradicar la duplicación de código heredada de mantener por separado las versiones de Coche y Moto. La solución se apoya en un patrón de **componente compartido alimentado por wrappers finos** y en la externalización de todo el contenido editable a una **capa de datos** (src/data/) que opera como un gestor de contenidos estático, de modo que el equipo de Marketing pueda mantener la web sin escribir código.

El **segundo subsistema** es el motor de inteligencia de telemetría. Su núcleo metodológico es una **arquitectura dual Box/Edge**: en el garaje (Box) se despliega un perceptrón multicapa (Multi-Layer Perceptron, MLP) de alta capacidad para el análisis fino tras la sesión, y en el volante (Edge) un árbol de decisión ligero, causal e interpretable, capaz de inferir en milisegundos sobre una Raspberry Pi y de tolerar la pérdida intermitente de sensores. Ante la ausencia de un histórico denso de telemetría del propio monoplaza, se emplea como **banco de pruebas metodológico** la telemetría pública de Fórmula 1 servida por la librería *fastf1* [1]; el problema concreto que se modela es la **clasificación de la marcha engranada** a partir de la telemetría instantánea, por ser una función física bien definida del estado del tren motriz y, por ello, un excelente termómetro de si el modelo aprende una ley transferible o memoriza un trazado. El hilo conductor de este subsistema es la **ingeniería de datos rigurosa**: la prevención de la fuga de información (data leakage) mediante una validación que respeta la estructura temporal y espacial del problema, y la garantía de causalidad de cada variable para que la inferencia sea válida en el Edge.

1.4. Estructura de la memoria

La memoria se organiza conforme a la estructura normalizada de la titulación. El Capítulo 2 describe las tecnologías empleadas en ambos subsistemas, el stack de desarrollo web frontend y los fundamentos de la adquisición, el modelado y el despliegue embarcado de la telemetría, y sienta su base teórica. El Capítulo 3 revisa el estado de la cuestión, webs de equipos de competición, análisis de telemetría en motorsport, inferencia embarcada y honestidad metodológica, y precisa la aportación diferencial del trabajo. El Capítulo 4 define formalmente el trabajo: justificación, objetivos, metodología, planificación y estimación económica unificadas. Los dos capítulos siguientes detallan los subsistemas: el Capítulo 5 desarrolla la **plataforma web**, análisis, diseño, implementación y gestión de contenido, y el Capítulo 6 el **sistema de inteligencia de telemetría Box/Edge**. El Capítulo 7 presenta y discute los resultados de ambos subsistemas, y el Capítulo 8 recoge las conclusiones y las líneas de trabajo futuro que cierran el ciclo de extremo a extremo. Finalmente, el Anexo A analiza la alineación del proyecto con los Objetivos de Desarrollo Sostenible (ODS) y el Anexo B reúne el material complementario, incluidos los fragmentos de código de ambos subsistemas

que sostienen las decisiones de ingeniería y los enlaces a los repositorios públicos con el código completo.

Capítulo 2

Descripción de las Tecnologías

Este capítulo describe las tecnologías y los fundamentos teóricos sobre los que se sustentan los dos subsistemas del proyecto. Se agrupan en cuatro bloques: el stack de desarrollo web frontend de la plataforma de datos del equipo; la adquisición de telemetría; el stack de aprendizaje automático con la base teórica de los algoritmos empleados; y el paradigma de Edge Computing que condiciona el despliegue embarcado.

2.1. El stack de la plataforma web

El primer subsistema (la plataforma web de datos del equipo) se construye sobre un stack de desarrollo frontend moderno. Conviene situar sus tecnologías en el panorama actual antes de abordar el diseño concreto en el Capítulo 5.

2.1.1. Aplicaciones de página única (SPA)

El desarrollo web ha evolucionado desde el modelo clásico de páginas servidas íntegramente por el servidor hacia el paradigma de la **aplicación de página única** (Single Page Application, SPA). En una SPA el navegador carga una única página HTML y, a partir de ahí, el contenido se actualiza dinámicamente mediante JavaScript sin recargar la página completa al navegar, lo que se traduce en transiciones instantáneas y en una experiencia próxima a la de una aplicación de escritorio, lo que mejora la experiencia de uso. La web de ISC es una SPA con un único punto de entrada (`index.html`) y veinticuatro rutas gestionadas en el lado del cliente. La contrapartida habitual de las SPA, un peor posicionamiento en buscadores por la ausencia de HTML prerrenderizado, se mitiga aquí porque el tráfico procede mayoritariamente de redes sociales y de contactos directos con patrocinadores, más que de búsqueda orgánica.

2.1.2. React 19 y el paradigma de componentes

React [3] es una biblioteca de JavaScript para construir interfaces de usuario basada en **componentes**: piezas autocontenidas, reutilizables y componibles que reciben datos de entrada (props) y devuelven una descripción declarativa de lo que debe mostrarse. Este modelo encaja de forma natural con el problema del proyecto: si las divisiones de Coche y Moto comparten estructura y solo difieren en el contenido, un único componente bien diseñado puede representar ambas sin más que recibir distintas props. El proyecto emplea **React 19**, cuya unidad de composición es el componente funcional, una función de JavaScript que devuelve JSX, la extensión sintáctica que permite escribir marcado similar a HTML dentro del código. La filosofía de «componente compartido + wrappers finos» que vertebraba la plataforma (Sección 5.2.1) es, en esencia, una aplicación rigurosa de los principios de composición y reutilización de React.

Los Hooks son funciones especiales que dotan de estado y de efectos a los componentes funcionales sin recurrir a clases. El proyecto usa intensivamente los tres fundamentales: `useState` (estado local reactivo, p. ej. qué menú está abierto), `useEffect` (efectos sincronizados con el ciclo de vida, como registrar y limpiar observadores) y `useRef` (referencias mutables a nodos del DOM o a valores persistentes entre renderizados). Una apuesta de rendimiento relevante es el uso de la API nativa del navegador **IntersectionObserver** para las animaciones de aparición ligadas al scroll: en lugar de escuchar continuamente el evento de scroll, práctica costosa que degrada la fluidez, `IntersectionObserver` notifica de forma eficiente cuándo un elemento entra en el área visible, momento en el que se le añade una clase CSS que dispara su animación. Este patrón, repetido en casi todas las secciones de contenido y en los contadores numéricos animados, es un ejemplo de optimización consciente del consumo de recursos.

2.1.3. Vite, React Router y SCSS con sistema de tokens

El entorno de desarrollo y compilación es **Vite** [4], una herramienta de nueva generación que destaca por la velocidad de su servidor de desarrollo, sirve los módulos ES de forma nativa y emplea esbuild para transformaciones ultrarrápidas, con recarga en caliente casi instantánea, y por empaquetar la producción con Rollup. En este proyecto Vite cumple además un papel clave en la resolución de un **alias de rutas** (la arroba @ apuntando a src/), que elimina las rutas relativas frágiles. La navegación se gestiona con **React Router 7**, biblioteca estándar de facto que asocia componentes a rutas, incluidas rutas dinámicas con parámetros, como la ficha de un vehículo identificado por su id, y cuyo hook `useLocation` permite decisiones contextuales, como mostrar la barra de navegación del Coche o la de la Moto según la ruta.

Las hojas de estilo se escriben en **SCSS**, una extensión del CSS que añade variables, anidamiento, mixins y módulos. Su ventaja decisiva para un proyecto orientado a la coherencia de marca es la posibilidad de definir **tokens de diseño**: variables centralizadas

(`src/styles/_variables.scss`) que representan los colores corporativos, la tipografía y las medidas fundamentales, importadas mediante la regla `@use` en cada componente. Referirse siempre a estos tokens en lugar de a valores literales garantiza que un cambio de marca se propague de manera uniforme; como se verá en el Capítulo 7, la disciplina de no redefinir («sombrear») estos tokens fue una lección de arquitectura aprendida a raíz de un fallo de color real. La plataforma incorpora, además, librerías especializadas, Leaflet para los mapas interactivos de competición, Swiper para las galerías táctiles y react-pageflip para los boletines, y se despliega de forma continua en Vercel, que reconstruye y publica la web ante cada cambio del repositorio.

2.1.4. El contenido estático como gestor de contenidos ligero

Frente a un gestor de contenidos tradicional con base de datos y panel de administración, el proyecto adopta un enfoque de **contenido estático versionado**: todo el contenido editable se modela como objetos y arrays de JavaScript en la carpeta `src/data/`. La decisión es deliberada y ventajosa para el caso de uso: el contenido cambia con baja frecuencia (en su mayor parte una vez por temporada), no requiere autenticación ni permisos complejos, queda versionado en el control de código junto al resto del proyecto y elimina por completo la necesidad de mantener y asegurar un servidor. La contrapartida, que editar el contenido exige tocar ficheros del repositorio, se neutraliza con un manual operativo y con una estructura de datos tan legible que no exige conocimientos de programación. Esta elección es, conceptualmente, la misma disciplina de **separar el dato de la lógica** que rige el subsistema de telemetría, y constituye uno de los puentes metodológicos entre ambos.

2.2. Adquisición de telemetría

2.2.1. El bus CAN como origen del dato

En un monoplaza de competición la telemetría se origina en una red de sensores, velocidad de rueda, régimen del motor, posición del acelerador y del freno, temperaturas, tensiones de la batería, cuyos valores se transmiten por el bus CAN, un estándar de comunicación serie robusto y determinista diseñado para entornos de automoción [5]. El bus CAN ofrece tramas cortas, arbitraje por prioridad y alta inmunidad frente a la interferencia electromagnética, características críticas en un vehículo eléctrico donde el tren motriz, inversor, motor y batería de alta tensión, genera un entorno electromagnético especialmente hostil. La frecuencia de muestreo típica de los canales relevantes para la dinámica longitudinal se sitúa en el orden de las decenas de hercios, lo que genera series temporales densas y fuertemente autocorreladas, propiedad que, como se verá, condiciona de manera decisiva la estrategia de validación.

2.2.2. La telemetría de Fórmula 1 como banco de pruebas: `fastf1`

Dado que el equipo ISC no dispone todavía de un histórico denso de telemetría de su propio monoplaza, este trabajo emplea como banco de pruebas la telemetría pública de Fórmula 1 accesible mediante la librería `fastf1` [1], escrita en Python. Esta librería proporciona acceso programático a datos de sesión oficiales (tiempos, telemetría de canal y posición) con una frecuencia de muestreo elevada, e incorpora una capa de caché local que permite trabajar sin conexión una vez descargada la sesión. La elección se justifica porque la física de fondo, dinámica longitudinal, relación cinemática entre régimen de motor y velocidad de rueda, fases de aceleración y de frenada, es común a cualquier vehículo de competición, lo que hace que la metodología validada sobre estos datos sea trasladable al monoplaza eléctrico con un coste de adaptación reducido.

Las señales empleadas son la velocidad (km/h), el régimen del motor (RPM), la marcha engranada (`nGear`), la posición del acelerador (`Throttle`, en %), el freno, el estado del sistema de reducción de resistencia (`Drag Reduction System`, `DRS`) y la distancia recorrida en la vuelta. Es importante subrayar una decisión de ingeniería de la fase de adquisición: para desacoplar el pipeline de la disponibilidad y la latencia de los servidores remotos, se diseñó un extractor independiente que descarga las sesiones y materializa un fichero `telemetry_dataset.csv` con un esquema fijo, de modo que el cuaderno de análisis consume siempre una fuente local estable y reproducible. Esta separación entre extracción y análisis es buena práctica de ingeniería de datos: aísla el cómputo determinista de la dependencia de red y garantiza que dos ejecuciones del análisis partan exactamente del mismo dato.

2.3. Aprendizaje automático: fundamentos y stack

El modelado se desarrolla íntegramente en Python con la librería `scikit-learn` [2], estándar de facto para el aprendizaje automático clásico, apoyada en `pandas` para la manipulación tabular y en `Matplotlib` para la visualización. La elección de `scikit-learn` responde a tres criterios de ingeniería. En primer lugar, su **API uniforme** (`fit/predict/transform`) y el objeto `Pipeline`, que permite encadenar el preprocesamiento y el estimador en una única entidad reajutable, condición necesaria para una validación cruzada sin fuga de información, pues garantiza que el escalado y cualquier transformación aprendida se ajusten solo con los datos de entrenamiento de cada partición. En segundo lugar, la disponibilidad de **validadores agrupados** (`GroupKFold`) que respetan la estructura del problema. En tercer lugar, la **frugalidad** de los modelos generados, que pueden serializarse y desplegarse en hardware limitado.

2.3.1. Formalización del problema de clasificación

El problema se formaliza como un aprendizaje supervisado multiclase. Dado un conjunto de muestras $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ con $\mathbf{x}_i \in \mathbb{R}^d$ el vector de características de telemetría e $y_i \in \{1, \dots, 8\}$ la marcha engranada, se busca una función $f: \mathbb{R}^d \rightarrow \{1, \dots, 8\}$ que minimice el riesgo esperado. Dado el fuerte desequilibrio de clases (ciertas marchas dominan el conjunto), la métrica de exactitud (accuracy) se complementa con la **F1 ponderada**, media armónica de precisión y exhaustividad (recall) promediada por el soporte de cada clase:

$$F1 = 2 \frac{\text{precisión} \cdot \text{recall}}{\text{precisión} + \text{recall}}, \quad \text{recall}_c = \frac{VP_c}{VP_c + FN_c}, \quad (2.1)$$

donde VP_c y FN_c son los verdaderos positivos y falsos negativos de la clase c . Reportar ambas métricas evita que un buen resultado en las clases mayoritarias enmascare el fracaso en las minoritarias, en este caso, las marchas poco frecuentes pero relevantes para la gestión del tren motriz.

2.3.2. Árbol de decisión

El árbol de decisión es un clasificador que particiona recursivamente el espacio de características mediante umbrales sucesivos sobre variables individuales, formando fronteras ortogonales a los ejes. En cada nodo se elige el par (variable, umbral) que maximiza la reducción de impureza. La impureza de Gini de un nodo t con proporciones de clase p_c es:

$$G(t) = 1 - \sum_{c=1}^C p_c^2, \quad (2.2)$$

y la división escogida es la que maximiza la disminución ponderada $\Delta G = G(t) - \frac{N_L}{N} G(t_L) - \frac{N_R}{N} G(t_R)$. Sus virtudes para el Edge son decisivas: es **interpretable** (cada predicción es una secuencia de comparaciones legibles), **determinista**, **barato** en inferencia y directamente **transpilable** a una función if/else en C, lo que permite eliminar por completo la dependencia del intérprete de Python en producción. Su principal limitación, fronteras escalonadas que aproximan mal las transiciones suaves, es justamente la que el MLP del Box compensa.

2.3.3. Perceptrón multicapa (MLP)

El perceptrón multicapa es una red neuronal de propagación hacia delante que compone transformaciones afines y no linealidades. Para una capa l :

$$\mathbf{h}^{(l)} = \sigma(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad (2.3)$$

con σ la función de activación ReLU ($\sigma(z) = \max(0, z)$). Gracias a estas capas ocultas, el MLP aproxima **fronteras de decisión no lineales y suaves**, lo que le permite resolver mejor las zonas ambiguas, las transiciones de cambio de marcha, donde dos marchas comparten un rango de régimen por la histéresis del cambio. Esta capacidad tiene un coste: mayor número de parámetros, mayor huella de memoria y entrenamiento más costoso, lo que lo hace idóneo para el Box pero inadecuado para el volante.

2.3.4. Técnicas no supervisadas: PCA y K-Means

El análisis de componentes principales (Principal Component Analysis, PCA) es una técnica de reducción de dimensionalidad que proyecta los datos sobre las direcciones ortogonales de máxima varianza, esto es, sobre los autovectores de la matriz de covarianza $\Sigma = \frac{1}{N} \mathbf{X}^T \mathbf{X}$ (con los datos centrados y escalados). En este trabajo el PCA cumple un doble papel: reducir dimensionalidad y, sobre todo, **interpretar físicamente** el espacio de características, pues cada componente principal admite una lectura como modo de operación del vehículo. El algoritmo K-Means, que agrupa las muestras minimizando la inercia intracluster $\sum_k \sum_{\mathbf{x} \in S_k} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2$, se emplea sobre el espacio reducido para detectar **regímenes de conducción** (recta a fondo, frenada fuerte, curva lenta, tracción de salida), etiqueta que se inyecta como característica adicional en una variante «híbrida» del modelo.

2.4. Edge Computing: la Raspberry Pi del volante

El Edge Computing es el paradigma según el cual el procesamiento de los datos se realiza en el propio extremo de la red (el dispositivo que los genera) en lugar de enviarlos a un servidor remoto [6]. En un vehículo de competición esta arquitectura es la única viable para las decisiones de control en tiempo real: la latencia, la disponibilidad y el determinismo no toleran la dependencia de un enlace de comunicaciones externo. La plataforma de referencia para la inferencia embarcada en este trabajo es la **Raspberry Pi**, un ordenador de placa única de bajo coste y consumo reducido, suficiente para ejecutar un modelo ligero pero con recursos de memoria y de cómputo muy inferiores a los de un servidor.

Esta restricción de hardware impone tres requisitos no funcionales sobre el modelo embarcado, que condicionan todo el diseño. Primero, la **causalidad**: toda variable de entrada debe poder calcularse a partir únicamente del pasado y del instante actual, pues en el coche el futuro no existe. Segundo, la **latencia acotada**: la inferencia debe completarse dentro del presupuesto temporal del lazo de control, típicamente 10 ms para un lazo a 100 Hz. Tercero, la **frugalidad**: el modelo debe ocupar poca memoria y depender de pocas señales para minimizar la superficie de fallo, cada sensor adicional del que depende el modelo es un punto de fallo más en un entorno donde la pérdida intermitente de señales no es excepcional, sino esperable. Estos tres requisitos motivan, conjuntamente, la asignación del árbol de decisión

(y no del MLP) a la unidad del volante, y se retomarán cuantitativamente en el Capítulo 7.

Capítulo 3

Estado de la Cuestión

Coherentemente con el carácter dual del proyecto, este capítulo revisa el estado de la cuestión en los dos dominios que confluyen en la plataforma de datos del equipo: las plataformas web de los equipos de competición, por un lado, y el análisis de telemetría, la inferencia embarcada y la honestidad metodológica en aprendizaje automático, por otro. La sección final precisa la aportación diferencial del trabajo en su conjunto.

3.1. Plataformas web de equipos de competición

La presencia digital se ha consolidado como un activo estratégico de los equipos de Formula Student punteros, que emplean su web como una herramienta de comunicación de ingeniería orientada a la captación de talento y de patrocinio. Como parte del análisis inicial de este trabajo se estudiaron las plataformas de varios equipos de referencia internacional, entre ellos AMZ Racing (ETH Zúrich) [7] y Formula Student Team Delft [8], de las que se extrajeron los patrones recurrentes que estructuran una web percibida como profesional: una portada con un impacto visual fuerte (vídeo o fotografía a pantalla completa y una llamada a la acción clara); una presentación del producto técnico (fichas de vehículo con especificaciones, galería e historial); una sección de captación de patrocinadores articulada por niveles (tiers) con una propuesta de valor explícita y un dossier descargable; y un organigrama que transmite el tamaño y la seriedad del equipo. Todos estos patrones se incorporan a la plataforma de ISC, adaptados a su doble naturaleza Coche/Moto.

Desde el punto de vista de la **ingeniería del software**, el reto recurrente en estas plataformas es la **gestión de la variación**: un mismo equipo con varias divisiones, o con vehículos de evolución anual, necesita el mismo conjunto de páginas con distinto contenido. La práctica menos costosa a corto plazo (duplicar el código por cada variante) degenera en una deuda técnica que multiplica el mantenimiento y propicia la divergencia visual. La buena práctica que el desarrollo frontend moderno opone a este problema es la **composición de componentes parametrizados** y la **externalización del contenido** a una capa de datos, de modo que la

lógica se escriba una sola vez y el contenido se gestione de forma independiente. Este trabajo adopta y sistematiza dicha práctica como principio rector del subsistema web.

3.2. Análisis de telemetría en motorsport

El análisis de telemetría es una disciplina madura en la alta competición. Los equipos de Fórmula 1 y de resistencia operan salas de telemetría que reciben cientos de canales en tiempo real y los procesan con suites comerciales y herramientas propietarias de los fabricantes de unidades de control. Estas soluciones priorizan la cobertura funcional y la integración con el banco de ensayo, pero son cerradas, costosas y diseñadas para infraestructuras de gran capacidad de cómputo en el garaje, lejos de las restricciones de un equipo universitario. Su enfoque, además, es predominantemente **descriptivo**: dan soporte a que un ingeniero experto interprete las trazas, más que a que un modelo infiera o decida de forma autónoma.

En el ámbito de Formula Student, la práctica habitual consiste en registrar la telemetría mediante adquisidores comerciales y analizarla a posteriori con software de visualización que permite superponer vueltas y comparar trazas. El análisis es mayoritariamente descriptivo y manual: el ingeniero inspecciona las curvas y extrae conclusiones cualitativas. Equipos punteros como AMZ Racing (ETH Zúrich), pionero en propulsión eléctrica y conducción autónoma [7], o Formula Student Team Delft [8] han incorporado culturas de ingeniería intensivas en datos, pero la aplicación sistemática de aprendizaje automático para modelar el comportamiento del vehículo, y, más aún, para embarcar esa inferencia en tiempo real, sigue siendo poco frecuente en el día a día de la mayoría de los equipos. Cuando se aplica ML, casi siempre se concibe para ejecutarse en el garaje, no embarcado, lo que deja sin explotar el horizonte de la decisión en tiempo real.

3.3. Inferencia embarcada en tiempo real (Edge AI)

La inferencia de modelos de aprendizaje automático en el extremo (Edge AI) ha experimentado un fuerte impulso con la proliferación de dispositivos de bajo consumo y de frameworks de despliegue ligero. En el sector de la automoción y la robótica, la motivación es siempre la misma: las decisiones de control no pueden depender de la latencia ni de la disponibilidad de un enlace remoto, de modo que el modelo debe residir y ejecutarse en el propio vehículo [6]. Las técnicas habituales para hacer viable un modelo en el Edge incluyen la reducción de su tamaño (pruning, cuantización), la selección de modelos intrínsecamente ligeros (árboles, modelos lineales) frente a redes profundas, y la transpilación del modelo a código nativo para eliminar la sobrecarga del intérprete. La literatura coincide en que, para problemas donde la estructura del fenómeno es aproximadamente conocida, un modelo simple y auditable suele ser preferible en el Edge a una red profunda cuya ganancia marginal

de precisión no compensa su coste de cómputo, memoria y opacidad.

3.4. Honestidad metodológica: fuga de información y causalidad

Un aspecto crítico (y a menudo subestimado) de la inferencia embarcada sobre series temporales es la **causalidad de las características** y la prevención de la fuga de información (data leakage). El data leakage se define como la presencia, en el proceso de entrenamiento o validación, de información que no estaría disponible en el momento real de la predicción, y constituye una de las causas más frecuentes de modelos que prometen en el laboratorio y fracasan en producción [9]. En el análisis de telemetría, el leakage adopta dos formas especialmente insidiosas. La primera es la **fuga temporal**: emplear una validación que baraja aleatoriamente una serie autocorrelada hace que muestras casi idénticas (consecutivas en el tiempo) caigan simultáneamente en entrenamiento y en validación, de modo que el modelo «recuerda» en lugar de generalizar, y la métrica resultante está inflada. La segunda es la **fuga de características**: calcular una variable empleando información futura (un suavizado centrado que promedia muestras posteriores, o un umbral global calculado sobre todo el conjunto, test incluido) produce una variable que en tiempo real no puede construirse.

Numerosos trabajos que reportan métricas excelentes offline resultan inviables en producción exactamente por estas razones. La prevención del leakage y la garantía de causalidad son, por tanto, condiciones necesarias (no opcionales) para que una métrica de laboratorio tenga validez en el vehículo. Este trabajo asume esa exigencia como principio rector y, lejos de darla por supuesta, la **cuantifica**: mide explícitamente el sesgo que introduciría una validación incorrecta, convirtiendo una buena práctica habitualmente implícita en un resultado experimental defendible.

3.5. Aportación diferencial de este trabajo

Frente al panorama descrito, la aportación de este trabajo es, ante todo, de **integración**: aborda el ciclo de vida del dato del equipo de extremo a extremo, articulando bajo una misma estrategia la infraestructura web que capta y presenta el dato público y el motor de inteligencia que explota el dato de telemetría. En el subsistema web, la aportación es una arquitectura limpia que **erradica la duplicación** mediante el patrón de componente compartido con wrappers finos y una capa de datos que devuelve al equipo la autonomía para mantener su web. En el subsistema de telemetría, la propuesta presenta cuatro rasgos diferenciales. En primer lugar, una **arquitectura dual Box/Edge** explícita, que separa el modelo de alta capacidad para el análisis en el garaje del modelo ligero para la inferencia en tiempo real, asignando cada modelo al horizonte temporal y al hardware donde sus

propiedades resultan óptimas, y justificando esa asignación con medidas de precisión, latencia y huella de memoria. En segundo lugar, un **rigor metodológico demostrado, no afirmado**: el trabajo cuantifica el sesgo introducido por la fuga de información mediante una validación cruzada por circuitos (leave-one-circuit-out) y reserva un circuito completo (Mónaco) como test aislado, respondiendo a la pregunta honesta de si el modelo generaliza a un trazado nunca visto. En tercer lugar, una **ingeniería de características íntegramente causal**, verificada mediante la propiedad de invariancia de prefijo, que garantiza que el modelo del Edge es desplegable tal cual sobre la Raspberry Pi. En cuarto lugar, el carácter **trasladable** del pipeline al monoplaza eléctrico del equipo, que convierte el trabajo en una solución de extremo a extremo (adquisición, modelado honesto y despliegue embarcado viable) concebida para una necesidad real y con un coste material prácticamente nulo.

Capítulo 4

Definición del Trabajo

4.1. Justificación

La justificación del trabajo se articula en torno a una necesidad real del equipo ISC Racing Team que se manifiesta en los dos planos del dato descritos en el apartado 1.2. En el plano externo, el equipo arrastraba una plataforma web duplicada, acoplada e inconsistente, incapaz de proyectar la imagen profesional que exige la captación de patrocinio B2B y la atracción de talento; resolver esa deuda técnica y mejorar la presentación visual es una necesidad estratégica, pues de la imagen del equipo dependen recursos económicos y humanos. En el plano interno, el monoplaza eléctrico genera telemetría cuya explotación inteligente puede traducirse en ventaja competitiva, pero el equipo carece de una arquitectura que permita procesar esos datos y, sobre todo, llevar la inferencia al propio vehículo en tiempo real.

Abordar ambos planos en un mismo trabajo no es una mera acumulación de tareas, sino la respuesta coherente a una estrategia única de dominio del dato: el subsistema web construye la infraestructura de captación y presentación, y el subsistema de telemetría el motor de explotación, sobre una filosofía compartida de separación entre dato y lógica, frugalidad y autonomía. La construcción del segundo subsistema exige, además, resolver problemas no triviales de ingeniería de datos, prevención de fuga de información, causalidad de las características, validación honesta y despliegue en hardware limitado, que justifican un tratamiento riguroso y académicamente defendible.

A esta necesidad se suma una motivación metodológica. En el ecosistema de la competición universitaria, el análisis de telemetría es mayoritariamente descriptivo y los modelos de aprendizaje automático, cuando existen, se conciben para el garaje y rara vez se someten a una validación que respete la estructura temporal del dato. Existe, por tanto, un hueco para una solución que demuestre (con evidencia cuantitativa) que es posible construir un sistema analítico honesto y desplegable en el extremo con recursos modestos. La ausencia de un histórico de telemetría propio del equipo motiva, además, el uso de un banco de pruebas público (*fastf1*) que permita validar la metodología antes de trasplantarla, desacoplando el

desarrollo del calendario de adquisición de datos del propio monoplaza.

4.2. Objetivos

El objetivo general es **dotar a ISC Racing Team de una plataforma de datos de extremo a extremo**, desde la infraestructura web que capta y presenta el dato público hasta el motor de inteligencia que explota la telemetría del monoplaza. Este objetivo general se concreta en dos bloques de objetivos específicos, uno por subsistema.

El **primer bloque**, asociado a la plataforma web (Capítulo 5), persigue:

1. **Erradicar la duplicación de código** fusionando cada par de componentes Coche/Moto en un único componente compartido parametrizable, y **desacoplar la lógica de la presentación del contenido** editable mediante una capa de datos que actúe como gestor de contenidos estático.
2. **Elevar la experiencia de usuario y la interfaz a un nivel profesional** orientado a la captación de talento y de patrocinio B2B, garantizando la coherencia de marca mediante un sistema de tokens de diseño centralizado y un diseño responsive validado en móvil, y **dotar al equipo de un manual operativo** que le permita gestionar el contenido de forma autónoma.

El **segundo bloque**, asociado al sistema de inteligencia de telemetría (Capítulo 6), persigue cuatro objetivos encadenados de modo que cada uno habilita al siguiente:

3. **Construir un pipeline de telemetría reproducible y libre de fuga de información**, que abarque la adquisición, la limpieza por grupos, la ingeniería de características causales y la validación, y que sea directamente trasladable a los datos del monoplaza eléctrico del equipo.
4. **Diseñar y entrenar una arquitectura dual de modelos**: un MLP de alta capacidad para el análisis en el Box y un árbol de decisión ligero para la inferencia en tiempo real en el Edge, justificando cuantitativamente la asignación de cada modelo a su entorno mediante medidas de precisión, latencia y huella de memoria.
5. **Validar la capacidad de generalización del sistema** mediante una validación cruzada por circuitos (leave-one-circuit-out) y un test aislado sobre un circuito de régimen dinámico estructuralmente distinto (Mónaco), cuantificando además el sesgo que introduciría una validación incorrecta sobre la serie temporal autocorrelada.
6. **Demostrar la viabilidad de la inferencia embarcada**, evaluando la latencia, la huella de memoria y la robustez del modelo del Edge frente a los requisitos de un lazo de control en tiempo real sobre la Raspberry Pi, y contrastando experimentalmente

la hipótesis de que un subconjunto mínimo de señales crudas basta para resolver el problema.

4.3. Metodología

Cada subsistema emplea la metodología propia de su disciplina, pero ambas comparten un mismo principio: validar cada cambio antes de avanzar. El **subsistema web** se aborda con una metodología **incremental e iterativa**, organizada en seis fases sucesivas, análisis y planificación; refactorización arquitectónica (unificación de componentes); reestructuración de carpetas por dominios; calidad, responsive y documentación; ajustes de UI/UX y contenido; y separación de recursos y color, que permiten obtener pronto una versión funcional y refinarla de forma continua. El trabajo se gestiona con control de versiones Git, cuyo historial de commits refleja con fidelidad esta progresión y aporta trazabilidad y seguridad ante cualquier edición incorrecta del contenido. Cuando la compilación de producción no podía ejecutarse por la ausencia del binario nativo de Rollup, se adoptó una **verificación por capas**, compilación de SCSS con el compilador de Sass, transformación del JSX con esbuild y comprobación de sintaxis de los datos con Node, que garantiza la corrección del código aun cuando el empaquetado final se realice en Vercel.

El **subsistema de telemetría** emplea la metodología propia del método experimental en aprendizaje automático, organizada como un pipeline de cinco fases, adquisición, limpieza, ingeniería de características, validación cruzada y test aislado, en el que **cada decisión metodológica se somete a comprobación cuantitativa**.

Con carácter previo, se realizó un trabajo de **planificación con visión global**: se descompuso el proyecto en tareas, se establecieron dependencias y prioridades, y se ordenó el conjunto de forma que el avance fuera sostenido a lo largo del curso. El seguimiento se gestionó mediante un tablero Kanban en Trello, que permitió mantener la trazabilidad del estado de cada tarea y trabajar al día sin acumulación de carga. La planificación se complementó con un **ciclo de retroalimentación continuo**: reuniones periódicas con los directores del trabajo y con la dirección del equipo ISC Racing Team, que aportaron criterio académico y validación de los requisitos reales del equipo en cada etapa, y que permitieron corregir el rumbo de forma temprana, en particular, en la detección y corrección de los errores metodológicos que se documentan en el Capítulo 6.

El principio rector de la fase experimental es la **honestidad estadística**. La partición de los datos respeta la estructura temporal y espacial del problema (agrupación por circuito y por vuelta) para no incurrir en fuga de información, y la capacidad de generalización se mide sobre un circuito que el modelo nunca ha visto. Las características se diseñan para ser causales y se verifica su validez antes de modelar. La comparación entre modelos se realiza bajo idénticas condiciones de validación, y las hipótesis de diseño, por ejemplo, la suficiencia de dos variables crudas, se contrastan mediante experimentos controlados. La

reproducibilidad se garantiza fijando las semillas aleatorias y serializando los parámetros de inferencia congelados, de modo que el artefacto desplegable sea idéntico al validado.

4.4. Planificación temporal

El trabajo se distribuyó a lo largo del curso académico 2025–2026 siguiendo de forma natural la estructura de los dos subsistemas, uno por cuatrimestre. Durante el **primer cuatrimestre** se desarrolló íntegramente la **plataforma web**, desde el análisis y el benchmarking inicial hasta la refactorización arquitectónica, la reorganización por dominios, la implementación de las secciones y la fase de calidad, responsive y documentación. Durante el **segundo cuatrimestre** se abordó el **subsistema de inteligencia de telemetría**, estado de la cuestión, adquisición y limpieza, ingeniería de características causales, validación cruzada, entrenamiento de los modelos Box/Edge e interpretabilidad, y, en paralelo, se aplicaron los **últimos detalles y ajustes finales a la web**; la redacción de la memoria se solapó deliberadamente con la fase experimental, de modo que las decisiones se documentaran en caliente. La figura 4.1 recoge esta planificación en forma de diagrama de Gantt.

El seguimiento se gestionó mediante un tablero Kanban en Trello, que mantuvo la trazabilidad del estado de cada tarea, y mediante un ciclo de retroalimentación continuo con los directores del trabajo y con la dirección del equipo ISC Racing Team. El hito central del segundo cuatrimestre, marcado en la figura, es la consolidación de un pipeline honesto y validado: hasta ese punto el esfuerzo se concentra en garantizar que la métrica es defendible (limpieza por grupos, características causales, detección y corrección de la fuga de información); a partir de él, en explotar esa base validada.

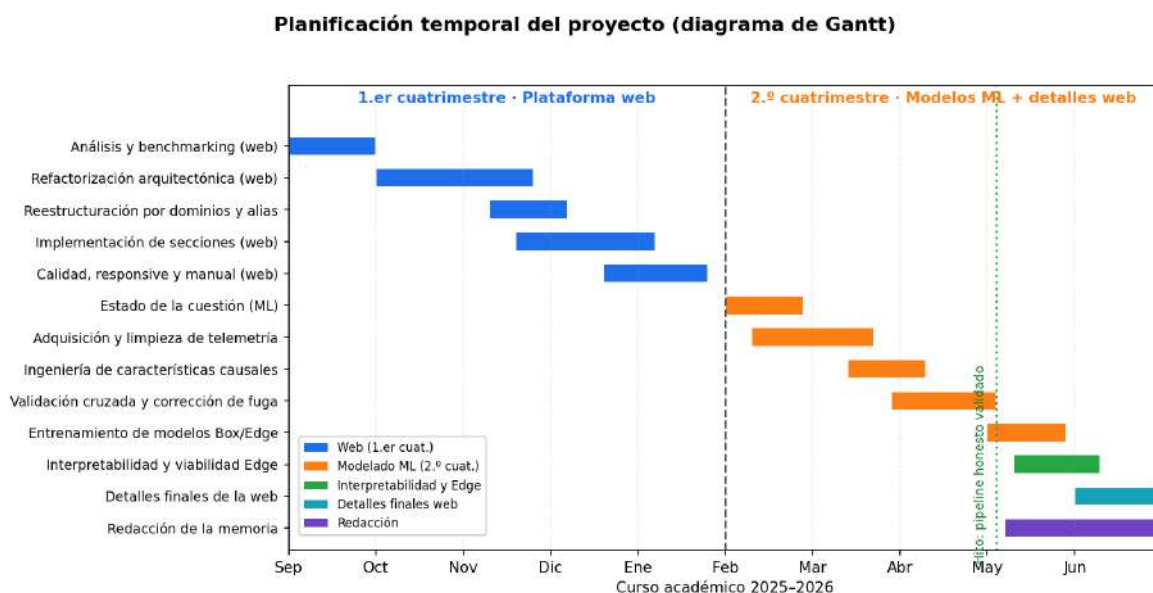


Figura 4.1. Diagrama de Gantt de la planificación del proyecto a lo largo del curso 2025–2026. El primer cuatrimestre (azul) se dedicó al desarrollo de la plataforma web; el segundo, al modelado de telemetría (naranja), a la interpretabilidad y viabilidad en el Edge (verde) y a los detalles finales de la web (turquesa), con la redacción de la memoria (morado) solapada. La línea de puntos marca el hito de validación del pipeline honesto.

La secuencia de tareas no es arbitraria, sino que refleja sus dependencias. En el subsistema web, la refactorización requería el análisis previo de la duplicación; en el de telemetría, la ingeniería de características causales no puede abordarse sin una limpieza previa por vuelta, la validación cruzada por circuitos carece de sentido sin características ya causales, y los experimentos de viabilidad en el Edge requieren modelos ya entrenados. Esta cadena de dependencias justifica el reparto por cuatrimestres y la posición del hito de validación como punto de no retorno del proyecto.

4.5. Estimación económica

La principal característica económica del proyecto es que **no supone ningún coste para ISC Racing Team**. Todas las decisiones de diseño se han tomado pensando en que el equipo no tenga que pagar nada, ni al desarrollarlo ni al mantenerlo. El trabajo lo ha realizado la autora como miembro del equipo, dentro de su labor de digitalización. El subsistema web se construye sobre tecnologías de código abierto (React, Vite, SCSS) y se publica de forma gratuita en Vercel, sin necesidad de un servidor propio ni de licencias. El subsistema de telemetría se apoya en datos públicos y en software libre, se desarrolla sobre un ordenador personal que ya se tenía y se prevé desplegar en una Raspberry Pi que el equipo ya posee. La tabla 4.1 resume estos recursos.

Tabla 4.1. Recursos del proyecto y su coste para el equipo.

Recurso	Solución adoptada	Coste
Desarrollo del trabajo	Realizado por la autora dentro del equipo	Gratuito
Tecnologías de la web	React, Vite, SCSS (código abierto)	Gratuito
Publicación de la web	Vercel (plan gratuito)	Gratuito
Datos y software de ML	Telemetría pública y librerías libres	Gratuito
Equipo de desarrollo	Ordenador personal ya disponible	Gratuito
Despliegue embarcado	Raspberry Pi de la que ya dispone el equipo	Gratuito
Licencias y servidores	No se necesitan	Gratuito

Que el proyecto sea gratuito es, en sí mismo, una aportación importante para el equipo. ISC Racing Team funciona con un presupuesto ajustado y cada euro cuenta para el coche y la moto. Conseguir una plataforma web profesional y un sistema de inteligencia de telemetría sin gastar nada permite dedicar todos los recursos económicos a la parte mecánica y electrónica de los vehículos. Además, como la solución no tiene gastos recurrentes (ni servidores, ni licencias, ni costes de mantenimiento externo), el equipo podrá seguir usándola y ampliándola en el futuro sin que ello suponga una carga económica.

Capítulo 5

Subsistema I: la plataforma web de datos

Este capítulo desarrolla el primer subsistema de la plataforma de datos de ISC: la web oficial del equipo, concebida como la infraestructura que capta, organiza y presenta el dato público (resultados, vehículos, organigrama, patrocinio) y que, en la visión de extremo a extremo del trabajo (apartado 1.2), constituye la superficie de comunicación y la futura capa de visualización de la inteligencia de telemetría. Se expone el análisis de requisitos, el diseño arquitectónico, la implementación, de forma condensada, remitiendo los listados completos al Anexo B, y el modelo de gestión de contenido para perfiles no técnicos.

5.1. Análisis

El punto de partida era una base de código que resolvía la dualidad Coche/Moto de la manera más directa y más costosa: **duplicando el código**. Existía un componente para el garaje del Coche y otro casi idéntico para el de la Moto, una barra de navegación por disciplina, una página de historia por duplicado, etcétera. Cualquier corrección debía aplicarse dos veces, con riesgo permanente de divergencia, y el contenido (textos y rutas de imágenes) vivía incrustado en los componentes de React, de modo que actualizar la descripción de un vehículo exigía editar código. A ello se sumaban inconsistencias de marca (verdes y tipografías dispares) y rutas relativas frágiles. El análisis sintetizó estas carencias en tres: duplicación de código, acoplamiento entre lógica y contenido, e inconsistencia visual.

Del análisis del equipo y del benchmarking (apartado 3.1) se derivaron los **requisitos funcionales** de la plataforma: una página de aterrizaje que bifurque al visitante hacia Coche o Moto; para cada disciplina, una portada multimedia, un garaje con fichas de todos los vehículos y una ficha de detalle por vehículo, una línea temporal histórica, un organigrama con filtrado por departamento, una zona de patrocinadores por niveles y una página de captación de patrocinio con dossier descargable, un mapa interactivo de competiciones y resultados, boletines de actualidad con visor de PDF integrado, y formularios de inscripción y contacto. Los **requisitos no funcionales**, coherencia de marca estricta, mantenibilidad

(ausencia de duplicación y separación lógica/contenido), rendimiento, gestión por perfiles no técnicos, diseño responsive y una presentación visual cuidada, son, en la práctica, los que más condicionan las decisiones de arquitectura.

La arquitectura de la información replica de forma simétrica las mismas secciones para ambas disciplinas, diferenciadas por el prefijo /moto en las rutas de la Moto, con la página de aterrizaje en la raíz. La tabla 5.1 recoge el mapa completo: veinticuatro rutas en total.

Tabla 5.1. Mapa de rutas de la plataforma (veinticuatro rutas en total).

Ruta (Coche)	Ruta (Moto)	Sección
/	—	Aterrizaje (Landing Split)
/formula	/moto	Portada (Home)
/history	/moto/history	Historia / Evolución
/garaje	/moto/garaje	Garaje (cuadrícula de vehículos)
/garaje/:id	/moto/garaje/:id	Detalle de vehículo
/team	/moto/team	Equipo / Organigrama
/resultados	/moto/resultados	Competiciones y resultados
/sponsors	/moto/sponsors	Patrocinadores
/sponsor-us	/moto/sponsor-us	Patrocínanos (B2B)
/articles	/moto/articles	Actualidad / Boletines
/recruitment	/moto/recruitment	Únete (recruitment)
/contact	/moto/contact	Contacto



Figura 5.1. Página de aterrizaje (Landing Split): bifurcación del visitante entre la división de Coche (Formula Student) y la de Moto (MotoStudent).

5.2. Diseño

El diseño descansa sobre tres principios estructurales que se aplican de forma transversal: el patrón de componente compartido con wrappers finos, la carpeta `data/` como fuente única de datos, y la organización por dominios.

5.2.1. El patrón componente compartido + wrappers finos

La pieza central de la arquitectura es el patrón que elimina la duplicación. Para cada funcionalidad existe un único **componente de presentación** en `components/data-display/` que no conoce nada concreto sobre el Coche o la Moto: recibe todo lo que necesita (textos, imágenes, rutas, datos) a través de sus props. A su vez, en `features/formula/` y `features/moto/` vive un **wrapper fino** por disciplina cuya única responsabilidad es importar el contenido adecuado y pasárselo al componente compartido. La diferencia de tamaño es ilustrativa: un wrapper como el garaje del Coche ocupa apenas una docena de líneas, mientras que el componente compartido concentra toda la lógica. El resultado es que cualquier mejora del garaje se implementa una sola vez y beneficia automáticamente a las dos disciplinas. El Listado 5.1 muestra el wrapper real del garaje del Coche: apenas una docena de líneas que importan los datos de los vehículos y se los pasan por props al componente compartido `GarageGrid`, que es quien contiene toda la lógica de presentación. El wrapper de la Moto es idéntico salvo en los datos que importa.

```

1 import GarageGrid from "@components/data-display/GarageGrid";
2 import { carsData } from "@data/formula/coches";
3
4 export default function Garage() {
5   return (
6     <GarageGrid
7       items={carsData}
8       subtitle="Pura ingenieria en numeros. Explora nuestra historia
9         tecnica."
10      basePath="/garaje"
11      itemNoun="Monoplaza"
12      ctaPrefix="SABER MAS DEL"
13      activeIds={["ifs08"]}
14    />
15  );
16 }

```

Listado 5.1. Wrapper fino del garaje del Coche (`features/formula/Garage.jsx`).

5.2.2. La carpeta `data/` como gestor de contenidos estático

El segundo principio es la **externalización del contenido**. Ningún texto, ruta de imagen o dato técnico vive incrustado en la lógica: todo reside en módulos de la carpeta `data/`, exportado como constantes legibles. Los componentes de presentación importan estos datos (o los reciben por props desde los wrappers) y se limitan a renderizarlos. Esta separación es la que habilita que un perfil no técnico actualice la web editando únicamente `data/` y `assets/`, sin riesgo de romper la lógica; cada fichero de datos incluye comentarios de cabecera que documentan la forma de cada registro, lo que lo convierte en una interfaz autoexplicativa. Conceptualmente, esta capa de datos es el análogo, en el subsistema web, de la separación entre dato y lógica que rige el pipeline de telemetría.

5.2.3. Normalización de rutas, organización por dominios y sistema de marca

Para eliminar las rutas relativas frágiles, el proyecto define un **alias de compilación**: la arroba `@` resuelve a la carpeta `src/`, declarado tanto en `vite.config.js` (para el empaquetador) como en `jsconfig.json` (para el editor y su autocompletado). El código se organiza **por dominios**: `features/` agrupa las páginas (landing, formula, moto), `components/` los compartidos (layout y data-display), `data/` el contenido editable, `styles/` los tokens y `assets/` los recursos por tipo. La coherencia visual se garantiza mediante un **sistema de tokens** centralizado en `_variables.scss` (tabla 5.2), de uso obligatorio en lugar de valores de color literales.

Tabla 5.2. Tokens principales del sistema de diseño (`_variables.scss`).

Token	Valor	Uso
<code>\$ib-main-green</code>	<code>#00402A</code>	Verde corporativo principal; navegación y CTA
<code>\$ib-sec-green</code>	<code>#00A443</code>	Verde secundario; acentos
<code>\$isc-yellow</code>	<code>#FFB81C</code>	Amarillo ISC; resaltados y subrayado activo
<code>\$car-orange</code>	<code>#FF7300</code>	Naranja asociado al Coche
<code>\$isc-grey</code>	<code>#5B5B5B</code>	Gris ISC; textos y detalles
<code>\$font-main</code>	<code>'Inter'</code>	Tipografía global
<code>\$max-width</code>	<code>1200px</code>	Anchura máxima del contenido

El Listado 5.2 reproduce la cabecera real del fichero de tokens. Cada componente importa estas variables con `@use` y se refiere a ellas por su nombre, de modo que un cambio de marca se propaga a toda la plataforma desde un único punto.

```
1 // Tipografia
2 $font-main: 'Inter', sans-serif;
```

```

3
4 // Colores corporativos ISC
5 $ib-main-green: #00402A; // verde principal (navegacion y CTA)
6 $ib-sec-green: #00A443; // verde secundario (acentos)
7 $isc-yellow: #FFB81C; // amarillo ISC (resaltados)
8 $scar-orange: #FF7300; // naranja del Coche
9 $isc-grey: #5B5B5B; // gris para textos y detalles
10
11 // Medidas
12 $max-width: 1200px;
13 $padding-section: 40px;

```

Listado 5.2. Cabecera de `src/styles/_variables.scss`: la paleta y la tipografía de marca.

Dado que el público objetivo accede mayoritariamente desde el móvil, el diseño **responsive** es un requisito de primer orden, validado mediante una auditoría específica. Complementariamente, se adoptó la técnica del **diseño para placeholders**, proporciones estrictas mediante `aspect-ratio` y `object-fit: cover`, para que la maqueta mantenga un aspecto profesional intacto aun antes de inyectar el material multimedia definitivo, de modo que la web se vea correcta tanto con las imágenes provisionales como cuando el equipo de Marketing las sustituya por las reales, sin reajustar el diseño.

5.3. Implementación

La base de código del frontend ronda las 7200 líneas. El arranque sigue la cadena habitual de React: `index.html` define el contenedor raíz y `main.jsx` monta la aplicación en modo estricto; `App.jsx` envuelve todo en el enrutador, decide qué barra de navegación y qué pie de página mostrar según la ruta, empleando `useLocation` para distinguir el dominio `/moto`, y declara las veinticuatro rutas. El Listado 5.3 reproduce el núcleo de `App.jsx`: la detección de la ruta actual, la barra de navegación y el pie de página que cambian según la disciplina, y la declaración de las rutas de cada división. Otros listados representativos (el patrón compartido mas wrapper, la configuración del alias y el contador animado) se reproducen en el Anexo B. A continuación se recorre cada sección de la plataforma con una breve descripción y una captura del resultado.

```

1 function AppContent() {
2   const location = useLocation();
3   const isLandingPage = location.pathname === "/";
4   const isMotoPage = location.pathname.startsWith("/moto");
5
6   return (
7     <div className="app">
8       <ScrollToTop />
9       {/* Navbar de la disciplina activa (Coche o Moto) */}

```

```

10      {!isLandingPage &&
11        (isMotoPage ? <Navbar menuItems={motoNav} />
12          : <Navbar menuItems={formulaNav} />)}
13
14      <Routes>
15        <Route path="/" element={<LandingSplit />} />
16        {/* Rutas del Coche (Formula Student) */}
17        <Route path="/formula" element={<Home />} />
18        <Route path="/garaje" element={<Garage />} />
19        <Route path="/garaje/:carId" element={<CarDetail />} />
20        <Route path="/team" element={<Team />} />
21        <Route path="/sponsors" element={<Sponsors />} />
22        <Route path="/sponsor-us" element={<SponsorUs />} />
23        {/* ... resto de rutas del Coche ... */}
24        {/* Rutas de la Moto (MotoStudent), prefijo /moto */}
25        <Route path="/moto" element={<MotoHome />} />
26        <Route path="/moto/garaje" element={<MotoGarage />} />
27        <Route path="/moto/garaje/:motoId" element={<MotoDetail />} />
28        {/* ... resto de rutas de la Moto ... */}
29      </Routes>
30
31      {!isLandingPage && <Footer />}
32      <ScrollToTopButton />
33    </div>
34  );
35 }

```

Listado 5.3. Nucleo de App.jsx: enrutado y layout segun la disciplina.

5.3.1. Navegación y pie de página

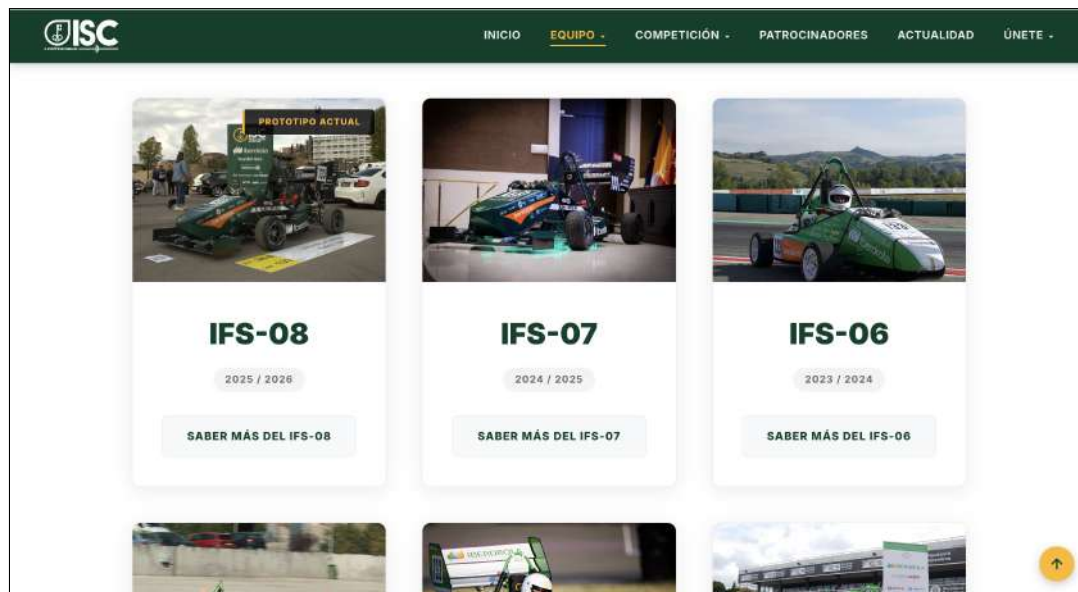
El layout compartido lo forman una barra de navegación (Navbar) y un pie de página (Footer). La Navbar es inteligente: con `useLocation` detecta si la ruta pertenece al Coche o a la Moto y muestra el menú y los colores de la disciplina activa, con desplegables para las secciones con subpáginas. El Footer es un componente parametrizable que recibe por props los enlaces, las redes sociales y los datos de contacto. Ambos se escriben una sola vez y sirven a las dos disciplinas.



Figura 5.2. Barra de navegación del Coche, con el menú y los colores de la disciplina activa, y pie de página con enlaces y redes sociales.

5.3.2. Portada (Home)

La portada abre con una imagen o un vídeo a pantalla completa y una llamada a la acción. Debajo se presentan el equipo, la misión y la visión, una galería y unos contadores numéricos que se animan al entrar en pantalla mediante `IntersectionObserver` (apartado 2.1.2). Es la página que marca el tono visual de toda la plataforma.



(a) Portada multimedia a pantalla completa.

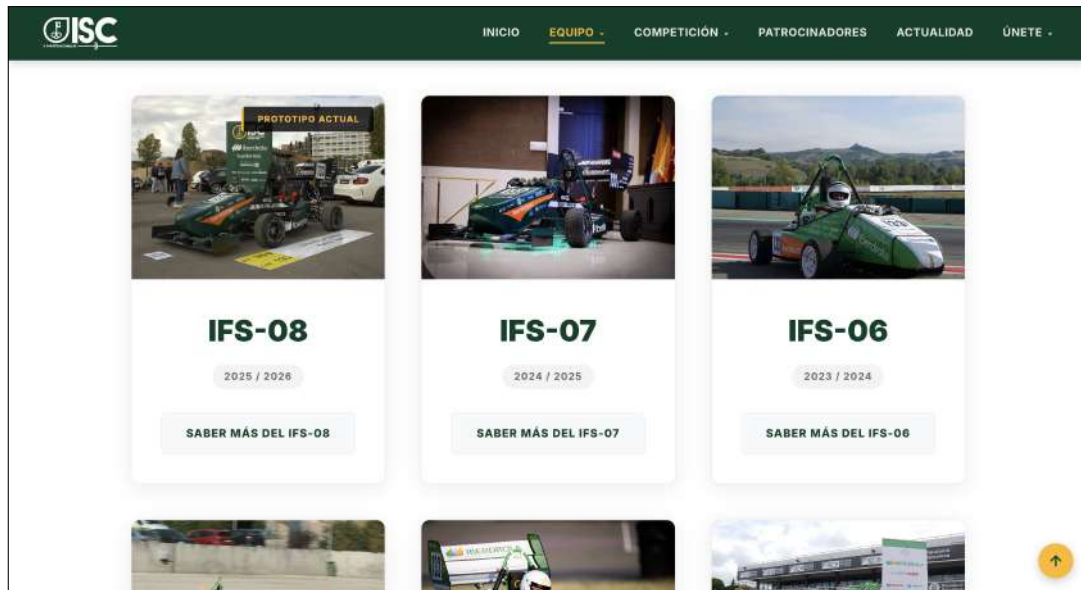


(b) Galería de imágenes y presentación del equipo.

Figura 5.3. Portada (Home) del Coche: portada multimedia a pantalla completa, presentación del equipo y contadores numéricos animados.

5.3.3. Garaje y ficha de vehículo

El garaje muestra una cuadrícula con todos los vehículos de la disciplina. Al pulsar en uno se abre su ficha de detalle, con las especificaciones técnicas, una galería de imágenes y el historial de competición. Tanto la cuadrícula como la ficha son componentes compartidos que reciben los datos del vehículo desde la capa data/.



(a) Cuadrícula de vehículos.



(b) Ficha de detalle con especificaciones, galería e historial.

Figura 5.4. Garaje: cuadrícula de vehículos, y ficha de detalle de un vehículo con especificaciones, galería e historial de competición.

5.3.4. Historia y línea temporal

La sección de historia presenta la evolución de los vehículos en una línea temporal. Cada hito incluye una imagen que puede ampliarse a pantalla completa con un visor (lightbox). El contenido de la línea temporal vive en `data/history.js`, de modo que añadir un nuevo año es tan sencillo como añadir una entrada a esa lista.

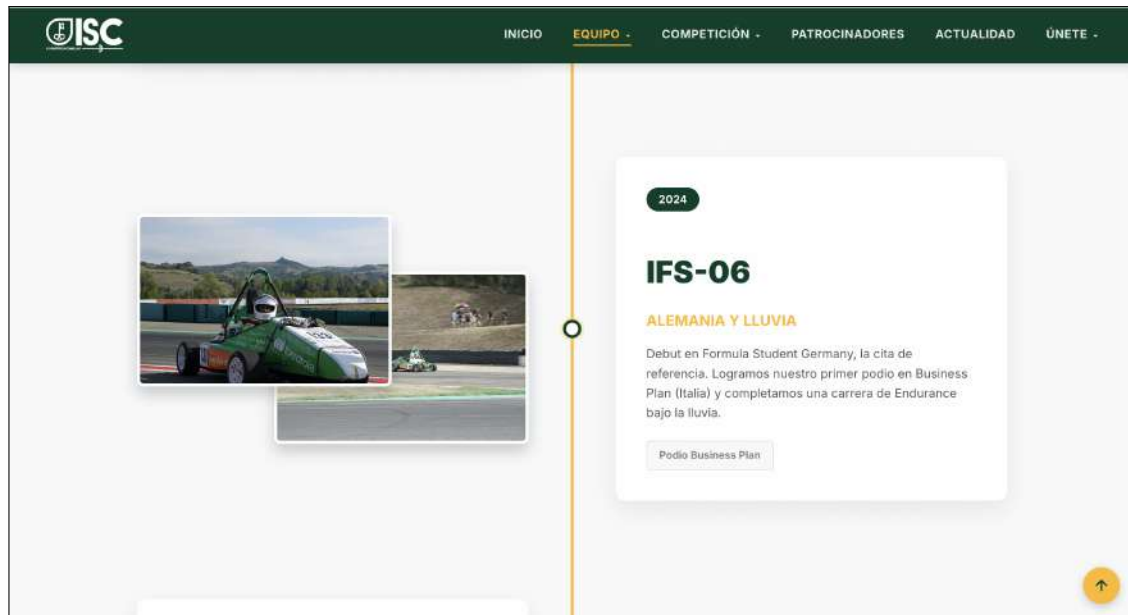


Figura 5.5. Historia: línea temporal con la evolución de los vehículos y ampliación de imágenes a pantalla completa.

5.3.5. Equipo y organigrama

El organigrama muestra a los miembros del equipo agrupados por departamento, con un filtro que permite ver solo un departamento concreto, y enlaces a los perfiles profesionales de cada persona. Transmite el tamaño y la seriedad del equipo, un factor valorado por los patrocinadores.

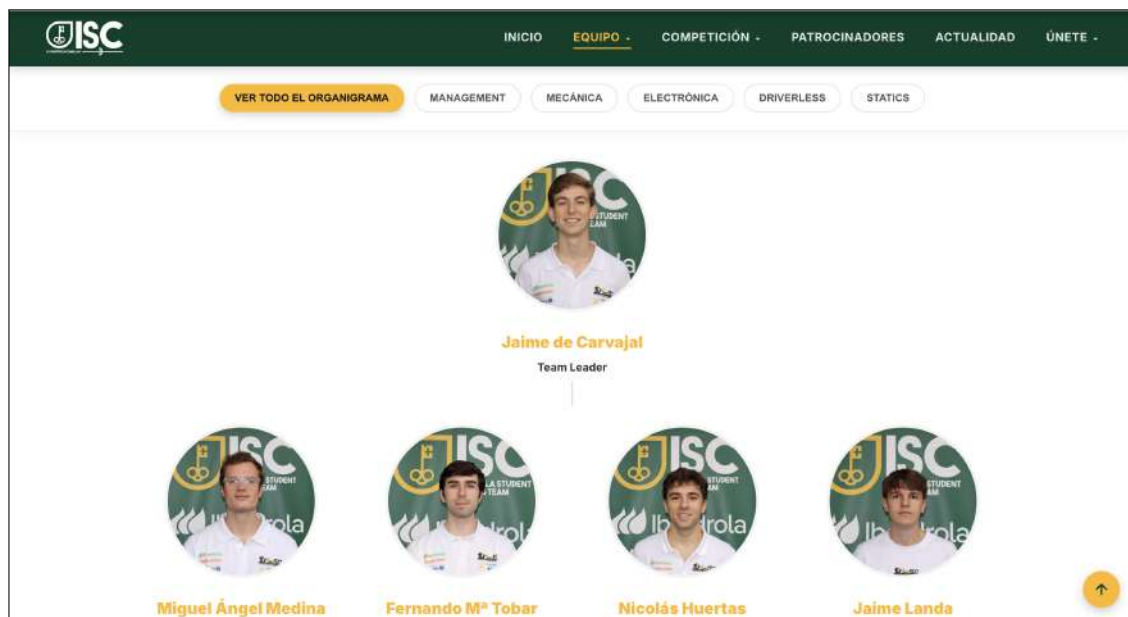


Figura 5.6. Organigrama del equipo con filtrado por departamento y enlaces a los perfiles profesionales de cada miembro.

5.3.6. Patrocinadores y captación de patrocinio

La plataforma dedica dos páginas al patrocinio. La de patrocinadores los muestra agrupados por niveles. La página «Patrocínanos» está pensada como una propuesta de captación de patrocinio (B2B): explica el valor de asociarse al equipo, presenta las categorías de patrocinio y permite descargar el dossier en PDF.

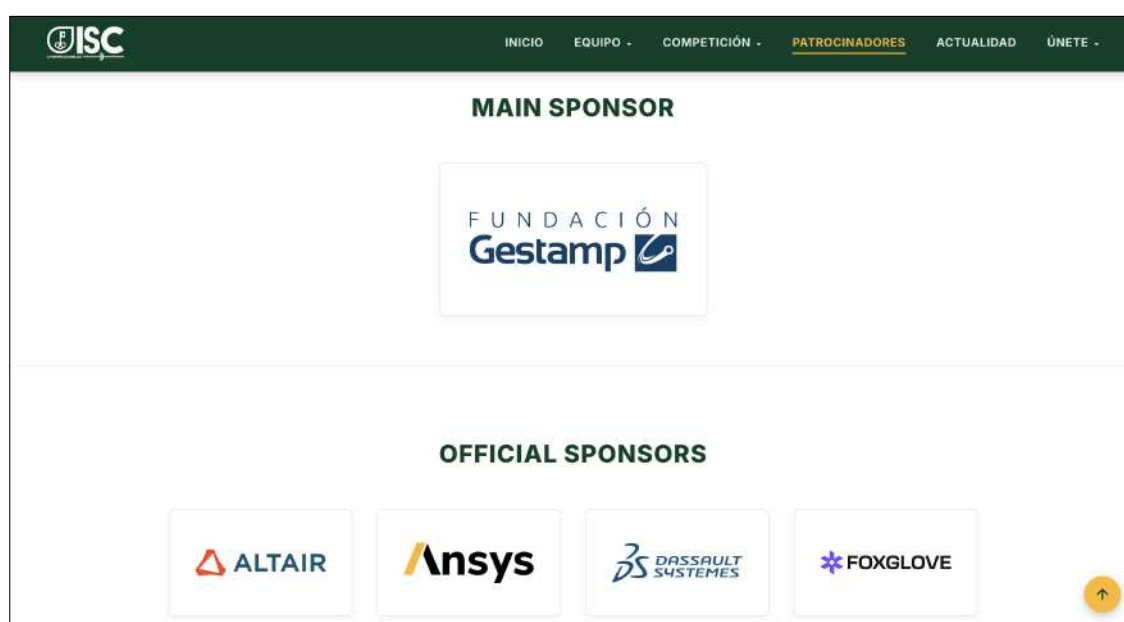


Figura 5.7. Patrocinadores agrupados por niveles.

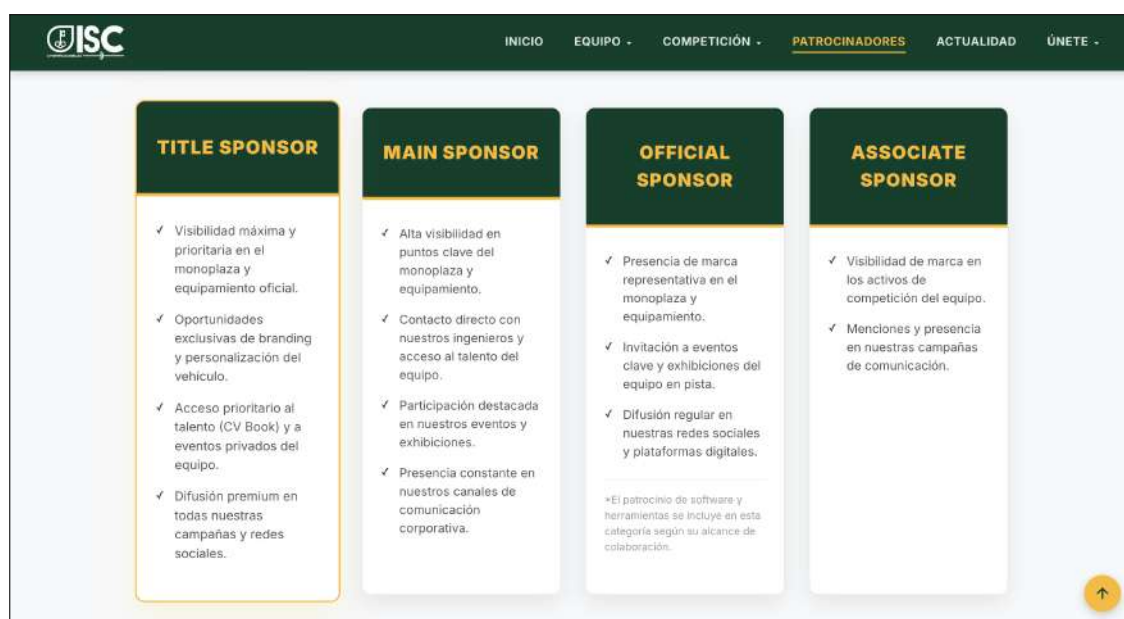
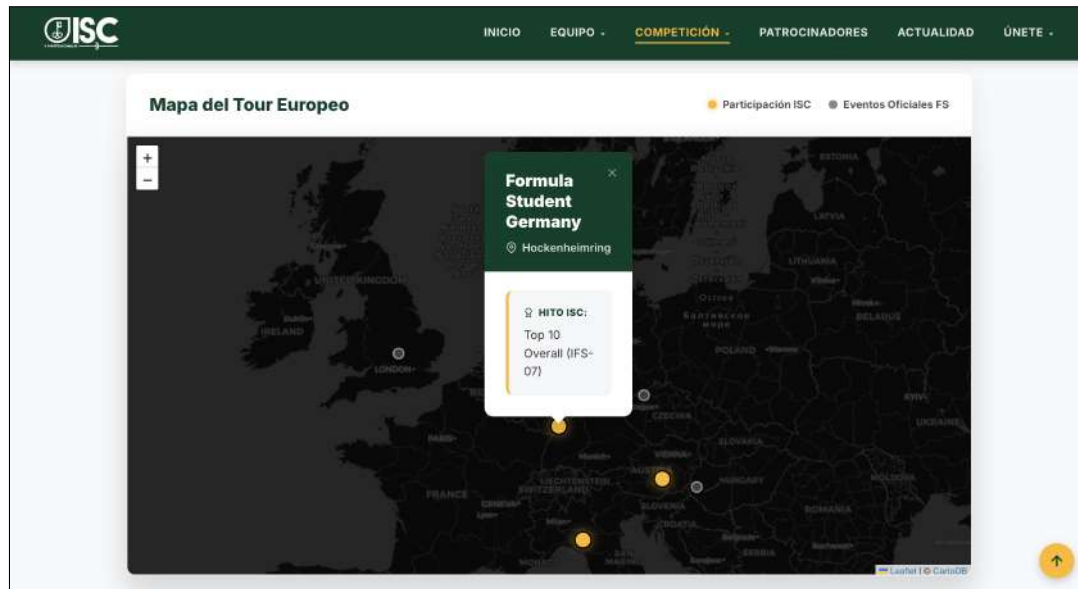


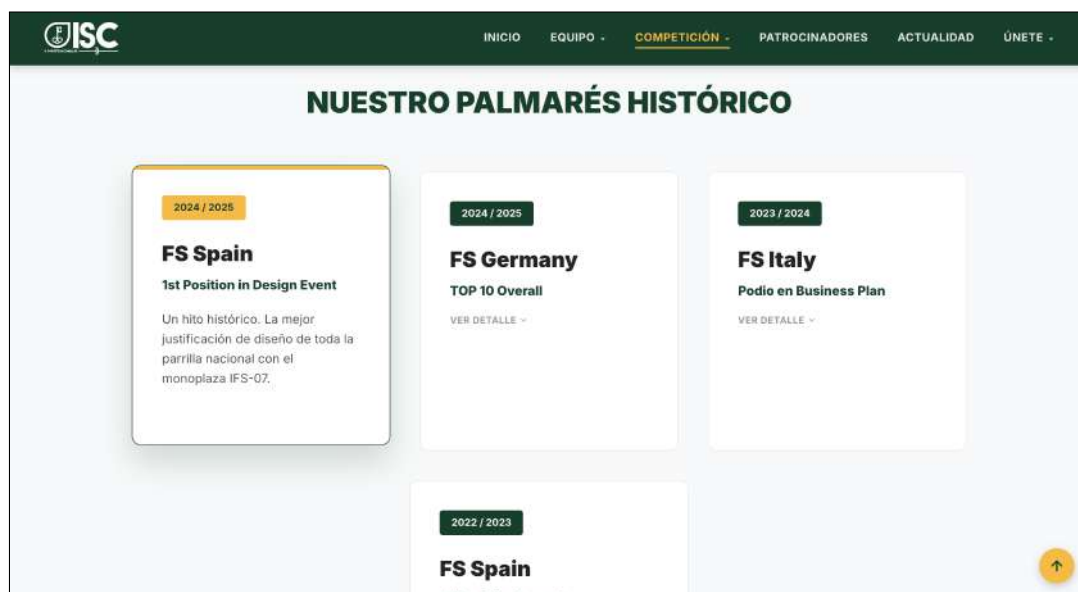
Figura 5.8. Página «Patrocínanos»: propuesta de valor B2B por niveles (tiers) con descarga del dossier de patrocinio.

5.3.7. Competiciones y mapa de resultados

Esta sección combina un mapa interactivo construido con Leaflet y el palmarés histórico del equipo. Sobre el mapa se sitúan marcadores que distinguen los eventos a los que el equipo ya ha asistido de los objetivos futuros, lo que da una imagen clara de su recorrido y sus metas.



(a) Mapa interactivo con marcadores de eventos asistidos y objetivos futuros.



(b) Palmarés histórico del equipo.

Figura 5.9. Competiciones: mapa interactivo de competiciones (Leaflet) con marcadores para eventos asistidos y objetivos futuros, junto al palmarés histórico.

5.3.8. Actualidad y boletines

La sección de actualidad reúne los boletines del equipo. Cada boletín se presenta con su portada y puede abrirse en un visor de PDF integrado, basado en un i frame que carga el

visor nativo del navegador, sin sacar al usuario de la web.



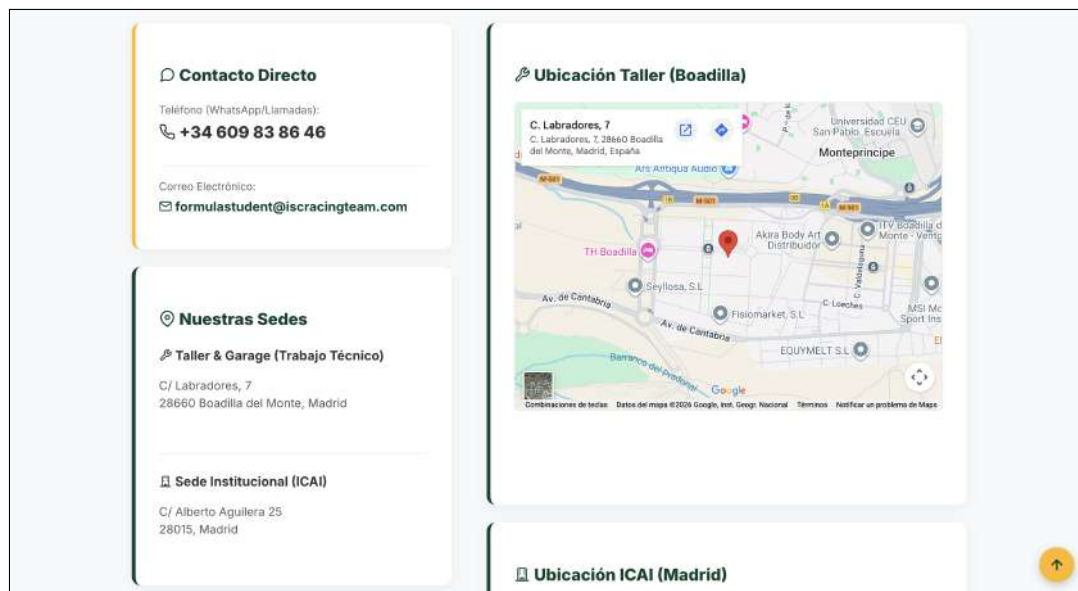
Figura 5.10. Actualidad: listado de boletines con su portada y visor de PDF integrado.

5.3.9. Únete y contacto

Las páginas de captación de talento (Únete) y de contacto cierran el recorrido. La primera presenta un formulario de inscripción para nuevos miembros; la segunda, un formulario de contacto junto a un mapa de ubicación y una sección de preguntas frecuentes.



(a) Página «Únete»: formulario de inscripción para nuevos miembros.



(b) Página de contacto con mapa de ubicación.

Figura 5.11. Páginas de captación de talento (Únete) y de contacto: formulario de inscripción, formulario de contacto y mapa de ubicación.

5.3.10. Diseño responsive

Como el público objetivo accede sobre todo desde el móvil, todas las secciones se han validado en pantallas pequeñas. La figura 5.12 compara la portada del Coche en escritorio y en móvil: el contenido se reordena y las imágenes mantienen sus proporciones gracias a las reglas de diseño descritas en la Sección 5.2.3.



Figura 5.12. Comparativa de la página de aterrizaje en escritorio y en móvil (diseño responsive): el contenido se reordena y las imágenes mantienen sus proporciones.

5.4. Gestión de contenido para perfiles no técnicos

Uno de los objetivos centrales es que el equipo de Marketing pueda mantener la web de forma autónoma, sin conocimientos de programación. Para ello el manual operativo (README_PARA_EL_EQUIPO.md) recurre a una **metáfora visual**: la carpeta `src` es un armario con cajones bien etiquetados, y solo dos de esos cajones conciernen al equipo, `data` (todo el contenido editable) y `assets` (el material gráfico y documental); los demás, donde vive la lógica, ni se abren. El manual documenta paso a paso las cuatro operaciones habituales (cambiar un texto, una fotografía, un vídeo o un documento descargable) indicando exactamente qué fichero tocar, y resume en una tabla del tipo «¿Dónde toco para. . . ?» la ubicación de cada tipo de cambio (tabla 5.3).

Tabla 5.3. Extracto de la tabla operativa del manual para Marketing.

Quiero cambiar. . .	Dónde se toca
La descripción de un coche	<code>src/data/formula/coches.js</code>
Un dato del organigrama	<code>src/data/team.js</code>
Un patrocinador o su nivel	<code>src/data/sponsors.js</code>
Un hito de la historia	<code>src/data/history.js</code>
Una foto existente	Reemplazar el archivo en <code>src/assets/</code> (mismo nombre)
Un boletín en PDF	<code>src/assets/articulos/</code> y <code>src/data/articles.js</code>

Cada disciplina cuenta con su propio contenido (textos, fotografías, vídeos y documentos), organizado en data y assets de forma separada, de modo que el Coche y la Moto son por completo independientes. El manual deja documentada la ubicación de cada recurso, lo que permite al equipo añadir o actualizar material en el futuro sin ayuda técnica. Este modelo de gestión, con el dato separado de la lógica, versionado y documentado para perfiles no técnicos, cierra el diseño del subsistema web y lo deja preparado para alojar, en su evolución futura, la visualización de la telemetría procesada por el subsistema de inteligencia que se describe en el capítulo siguiente.

Capítulo 6

Subsistema II: inteligencia de telemetría (Box/Edge)

El segundo subsistema es el motor de inteligencia de la plataforma de datos: el pipeline analítico de telemetría con arquitectura dual de modelos que convierte el dato crudo del monoplaza en predicciones. Mientras la plataforma web del Capítulo 5 capta y presenta el dato del equipo, este subsistema lo **explota**, en los dos horizontes temporales que vertebran el proyecto: el análisis reposado en el garaje (Box) y la inferencia embarcada en tiempo real sobre el vehículo (Edge).

6.1. Análisis

El sistema desarrollado es un pipeline analítico de telemetría con una arquitectura dual de modelos. Esta sección analiza sus requisitos antes de abordar el diseño y la implementación.

El problema se formaliza como una **clasificación supervisada multiclase**: predecir la marcha engranada (nGear, ocho clases) a partir de la telemetría instantánea del vehículo. La marcha se elige como variable objetivo por ser una función física bien definida del estado del tren motriz, concretamente, una función discreta de la relación entre régimen de motor y velocidad de rueda, lo que permite validar si el modelo aprende una ley transferible o memoriza particularidades del trazado. Esta elección convierte la tarea en un excelente banco de pruebas metodológico: un modelo que generalice bien sobre la marcha lo hará porque ha capturado la cinemática de la transmisión, no porque haya memorizado un circuito.

Los **requisitos funcionales** del sistema son: adquirir y preparar la telemetría de varios circuitos de forma reproducible; aprender la relación entre las señales instantáneas y la marcha; y producir dos artefactos de inferencia (uno para el Box y otro para el Edge). Los **requisitos no funcionales** más exigentes, derivados del despliegue embarcado descrito en la Sección 2.4, son tres y condicionan todo el diseño: la **causalidad** de las características, la **latencia acotada** de la inferencia y la **frugalidad** en memoria y en número de sensores.

A ellos se añade un requisito transversal de **honestidad estadística**: las métricas reportadas deben estar libres de fuga de información para ser defendibles.

6.2. Diseño

6.2.1. Arquitectura dual Box / Edge

El sistema no emplea un único modelo, sino **dos modelos con propósitos y restricciones opuestas**, asignados a dos entornos de ejecución distintos. La tabla 6.1 resume la dualidad.

Tabla 6.1. Arquitectura dual Box/Edge: dos cerebros para dos tiempos de respuesta.

	Box (garaje)	Edge (volante)
Hardware	Servidor / portátil	Raspberry Pi en el volante
Momento	Post-tanda, sin prisa	Tiempo real, milisegundos
Modelo	MLP profundo (alta cap.)	Árbol de decisión ligero
Prioridad	Máxima precisión	Latencia mínima, causalidad
Restricción	Ninguna relevante	Memoria, latencia, robustez

El **modelo del Box** (MLP) prioriza la capacidad de representación: opera tras la sesión, sin restricción de latencia, y se destina al análisis fino donde sus fronteras de decisión no lineales aportan precisión. El **modelo del Edge** (árbol de decisión) prioriza la viabilidad operativa: debe inferir en milisegundos, depender de pocas señales fiables y ser directamente auditable. Esta separación no es una concesión, sino la materialización correcta de un principio de ingeniería: desplegar cada modelo allí donde sus propiedades lo hacen óptimo.

6.2.2. Ingeniería de características causales

A partir de las señales crudas se derivan diez características diseñadas para ser **causales**, es decir, calculables a partir únicamente del pasado y del instante actual. Las principales son: la velocidad y el régimen del motor en crudo; el cociente `rpm_per_speed` (régimen por unidad de velocidad), que codifica directamente la relación cinemática de la transmisión; la aceleración longitudinal `a_lon`, calculada mediante derivada hacia atrás y media móvil sin centrar; un `power_proxy` (producto de régimen y acelerador) como aproximación a la demanda de potencia; e indicadores binarios de activación de DRS, rodadura inercial (coasting), zona de curva (`corner_zone`) y fase de aceleración.

Dos decisiones de diseño garantizan la causalidad. La primera afecta al cálculo de la aceleración longitudinal. La aceleración es la derivada de la velocidad, $a_{lon}(t) = dv/dt$, aproximada en tiempo discreto por una **diferencia hacia atrás** $(v_t - v_{t-1})/\Delta t$, que solo usa el pasado. Como esta derivada amplifica el ruido de la señal de velocidad, se suaviza con

una media móvil; pero el suavizado se realiza con una ventana **causal** (que solo promedia muestras pasadas) en lugar del suavizado centrado que utilizaba la versión inicial del pipeline y que promediaba muestras futuras. Es fundamental señalar que esta media móvil suaviza **únicamente** la señal ruidosa `a_lon`; la velocidad y el régimen entran en el modelo en crudo, sin filtrar. La segunda decisión afecta al umbral de `corner_zone`, que se **congela** a partir únicamente del conjunto de entrenamiento y se reaplica como constante en inferencia, en lugar de recalcularse con un cuantil global que incluiría los datos de test.

La figura 6.1 ilustra la diferencia y la propiedad de **invariancia de prefijo**: al truncar el futuro de la vuelta, la característica causal no cambia, mientras que la centrada sí lo hace. Esta propiedad es la verificación formal de la causalidad: una característica es desplegable en el Edge si, y solo si, su valor en el instante t es invariante ante la eliminación de toda muestra posterior a t .

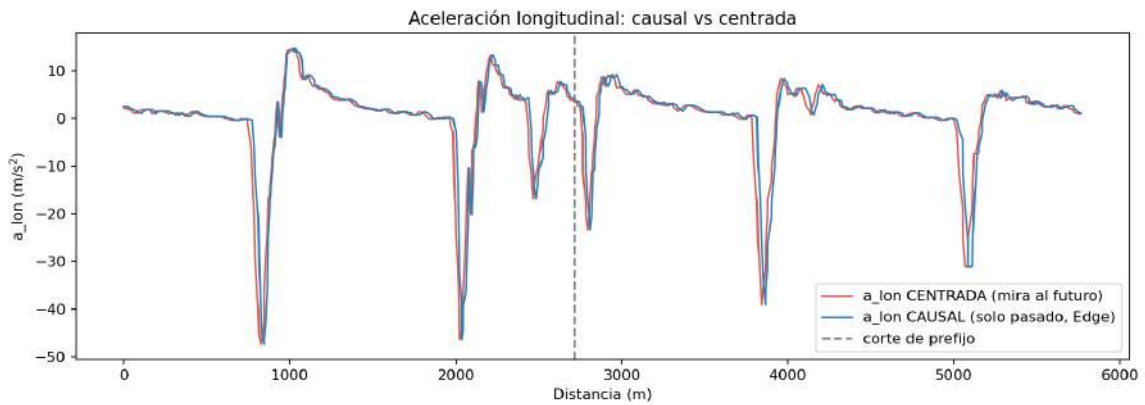


Figura 6.1. Aceleración longitudinal causal frente a centrada. La señal causal (azul) es invariante al truncar el futuro de la vuelta en el corte de prefijo, condición necesaria para su despliegue en el Edge; la centrada (rojo) usa información futura y cambia al truncar.

Adicionalmente, se incorpora un **detector de régimen de conducción** no supervisado: una reducción PCA seguida de una agrupación K-Means etiqueta cada muestra con el régimen en que se encuentra el coche (recta a fondo, frenada fuerte, curva lenta, tracción de salida). Esta etiqueta se inyecta como variable one-hot en la variante denominada «híbrida». Para evitar la fuga de información, el detector se encapsula en un transformador de scikit-learn que reajusta el PCA y el K-Means **dentro de cada partición** de la validación cruzada, y su predicción usa solo la muestra actual, por lo que es causal.

6.3. Implementación

6.3.1. Adquisición y limpieza

La **adquisición** carga la vuelta rápida de varios pilotos en cada circuito de entrenamiento (Monza, Spa, Silverstone) y etiqueta cada muestra con dos identificadores de grupo: el circuito

(grupo grueso) y la vuelta (grupo fino). Para desacoplar el análisis de la disponibilidad de los servidores remotos, la extracción se realiza con un script independiente que materializa un CSV de esquema fijo, que el pipeline consume como fuente local estable.

La **limpieza** se realiza por vuelta, de modo independiente para cada serie temporal, interpolando y rellenando los huecos dentro de cada grupo, nunca a través de la frontera entre vueltas, lo que mezclaría dinámicas distintas, y aplicando un filtro de rango físico absoluto (por ejemplo, velocidad ≤ 380 km/h, régimen $\leq 15\,500$ RPM). Este filtro es deliberadamente **absoluto y no estadístico**: al no depender de la distribución de los datos (de cuantiles o de medias del propio conjunto), no introduce fuga de información, a diferencia de un recorte basado en percentiles que «miraría» todo el conjunto, test incluido.

6.3.2. Validación cruzada por circuitos

El núcleo metodológico es la **validación cruzada por circuitos** (leave-one-circuit-out), implementada con GroupKFold agrupando por circuito: en cada partición se deja fuera un circuito completo y el modelo se valida sobre pista que no ha visto jamás. El circuito de **Mónaco** se reserva como test final aislado, fuera incluso de la validación cruzada.

Esta elección corrige el error de la versión inicial del pipeline, que empleaba una partición aleatoria barajada (StratifiedKFold con shuffle). Dado que la telemetría es una serie temporal autocorrelada, barajar provoca que muestras casi gemelas (consecutivas en el tiempo) caigan a la vez en entrenamiento y en validación, de modo que el modelo «reconoce» en validación muestras prácticamente idénticas a las que ha visto entrenando, inflando las métricas. La figura 6.2 cuantifica este sesgo comparando ambas particiones bajo idénticas condiciones, y constituye uno de los resultados metodológicos centrales del trabajo: convierte una buena práctica habitualmente implícita en una cifra defendible.

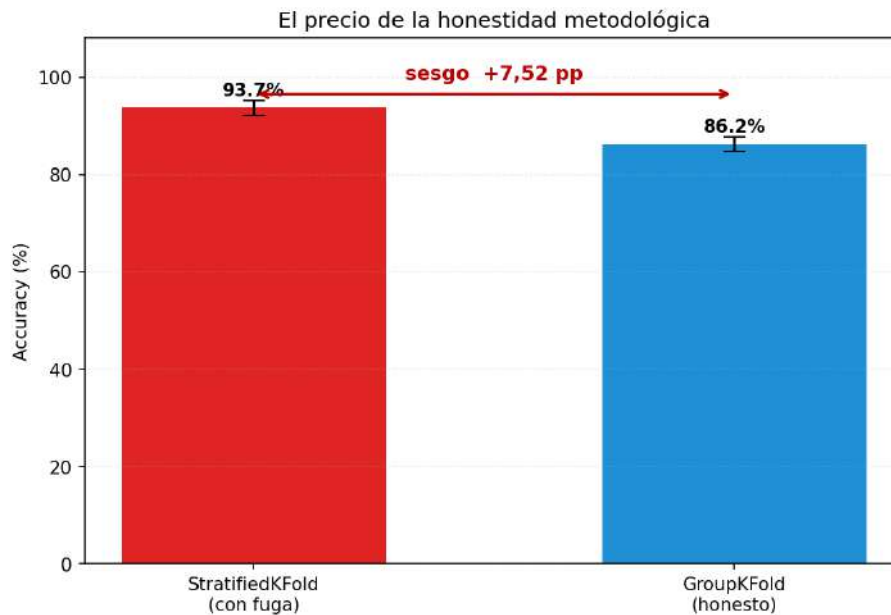


Figura 6.2. Cuantificación del sesgo por fuga de información. La partición barajada (con fuga) sobreestima la accuracy en 7,52 puntos frente a la partición honesta por circuitos.

Cada modelo se construye como un `Pipeline` de `scikit-learn` que encadena el escalado, el detector de régimen opcional y el estimador, mediante `factories` que garantizan una instancia limpia por partición, sin estado heredado. Dentro de cada partición, el escalado, el PCA y el K-Means se reajustan únicamente con los datos de entrenamiento de esa partición, nunca con los de validación, condición que el objeto `Pipeline` garantiza por construcción.

6.3.3. Modelos: MLP (Box) y Árbol de Decisión (Edge)

Ambos modelos comparten el mismo `Pipeline` de `scikit-learn` (Sección 6.3.2): un escalado estándar (`StandardScaler`), necesario para el MLP y neutro para el árbol; el detector de régimen opcional de la variante híbrida; y el estimador final. El umbral de `corner_zone` se congela con el percentil 33 de la velocidad de entrenamiento (221,5 km/h) y se reaplica idéntico en inferencia. El desequilibrio de clases se trata con `class_weight="balanced"` en el árbol; el MLP no admite ese parámetro, por lo que se entrena sin ponderación explícita y absorbe el desequilibrio mediante su capacidad y la parada temprana.

Búsqueda de hiperparámetros. Los hiperparámetros se eligieron mediante una **búsqueda en rejilla manual** con validación cruzada *leave-one-circuit-out* (`GroupKFold`, tres particiones por circuito), seleccionando por la exactitud media de validación cruzada. Para el **árbol del Edge**, con `min_samples_leaf = 10` y `class_weight = balanced` fijos, se barrió la profundidad (tabla 6.2). La profundidad 5 alcanza la exactitud máxima (87,13 %) pero con mayor varianza entre circuitos ($\pm 2,75$); se elige la profundidad 6 porque ofrece prácticamente la misma exactitud (86,18 %) con la menor varianza ($\pm 1,51$), es decir, mayor estabilidad

entre trazados, además de un árbol más ligero y auditable para el Edge.

Tabla 6.2. Barrido de profundidad del árbol del Edge (exactitud de validación cruzada *leave-one-circuit-out*).

max_depth	Exactitud CV (media ± desv.)
3	71,72 % ± 5,86
4	84,07 % ± 5,38
5	87,13 % ± 2,75
6 (elegido)	86,18 % ± 1,51
8	85,68 % ± 1,30
Sin límite	85,13 % ± 1,72

Para el **MLP del Box**, con `solver = adam`, `learning_rate_init = 10-3` y parada temprana fijos, se exploró la arquitectura de capas y la regularización `alpha` (tabla 6.3). La arquitectura (128, 64, 32) es la mejor de las profundidades de red evaluadas; con `alpha = 10-3` mejora marginalmente (88,99 %), pero la diferencia (0,75 puntos) queda dentro del ruido entre particiones, y ampliar a (256, 128, 64) no aporta mejora. Se conserva (128, 64, 32) con `alpha = 10-4` por parsimonia.

Tabla 6.3. Rejilla de arquitectura y regularización del MLP del Box (exactitud de validación cruzada *leave-one-circuit-out*).

hidden_layer_sizes	alpha	Exactitud CV (media ± desv.)
(64)	10 ⁻⁴	87,29 % ± 2,44
(64, 32)	10 ⁻⁴	87,87 % ± 2,35
(128, 64)	10 ⁻⁴	87,82 % ± 0,62
(128, 64, 32) (elegido)	10 ⁻⁴	88,24 % ± 2,44
(128, 64, 32)	10 ⁻³	88,99 % ± 2,06
(256, 128, 64)	10 ⁻⁴	87,80 % ± 2,34

Hiperparámetros finales. El **árbol del Edge** es un `DecisionTreeClassifier` con `criterion="gini"`, `max_depth = 6`, `min_samples_leaf = 10`, `class_weight="balanced"` y `random_state = 42`. El **MLP del Box** es un `MLPClassifier` con capas ocultas (128, 64, 32), activación ReLU, optimizador Adam, `alpha = 10-4`, `learning_rate_init = 10-3`, `batch_size = 64`, `max_iter = 500` y parada temprana (`validation_fraction = 0,1`, `n_iter_no_change = 15`).

Métricas de entrenamiento. La tabla 6.4 resume las métricas de entrenamiento y de validación cruzada de ambos modelos (la evaluación sobre el test aislado de Mónaco se

reserva para el Capítulo 7). El árbol alcanza una exactitud de entrenamiento del 90,91 % y el MLP del 91,76 %; la moderada distancia respecto a sus exactitudes de validación cruzada (86,18 % y 88,24 %) indica un sobreajuste contenido.

Tabla 6.4. Métricas de entrenamiento y de validación cruzada *leave-one-circuit-out* de ambos modelos.

Modelo	Exact. entren.	F1 entren.	Exact. CV	F1 CV
Árbol (Edge)	90,91 %	0,910	86,18 % $\pm$ 1,51	0,860 $\pm$ 0,020
MLP (Box)	91,76 %	0,918	88,24 % $\pm$ 2,44	0,879 $\pm$ 0,029

En el MLP, la **parada temprana** gobierna la duración del entrenamiento: el ajuste se detuvo en la época 47 de un máximo de 500. La pérdida de entrenamiento descendió de 1,155 (época 1) a 0,191 (época 47), y la exactitud sobre el subconjunto interno de validación pasó del 77,5 % inicial a un máximo del 92,0 % en la época 31; al no mejorar durante las 15 épocas siguientes, la parada temprana detuvo el ajuste, evitando el sobreajuste.

Un detalle de implementación relevante es el tratamiento del desequilibrio de clases. El árbol admite la ponderación mediante `class_weight`; el MLP no admite ese parámetro, de modo que el desequilibrio se gestiona con `sample_weight` en los estimadores que lo soportan y, en el MLP, mediante su capacidad y la parada temprana. Finalmente, se serializan los dos modelos y los **parámetros de inferencia congelados** (lista de características y umbral de `corner_zone`), lo que hace el despliegue reproducible y causal: la Raspberry Pi reaplica el mismo umbral aprendido en el Box, sin recalcularlo con datos nuevos.

Capítulo 7

Análisis de Resultados

Este capítulo presenta y discute los resultados de los dos subsistemas de la plataforma de datos. Primero se evalúan los resultados de la **plataforma web**, la reducción de la duplicación, un caso de depuración representativo y el grado de cumplimiento de los objetivos; a continuación, los resultados experimentales del **subsistema de inteligencia de telemetría**, con su arquitectura dual Box/Edge.

7.1. Resultados de la plataforma web

El resultado del primer subsistema es una plataforma web completa, coherente y mantenible que cubre las veinticuatro rutas previstas para las dos disciplinas. Cada sección funciona a partir de un único componente compartido y todo el contenido reside en una capa de datos editable de forma autónoma. La web presenta una presentación visual cuidada y coherente, con un sistema de marca uniforme, animaciones de aparición eficientes y un comportamiento responsive validado en móvil.

7.1.1. Métricas de código y reducción de la duplicación

La base de código del frontend ronda las 7200 líneas. Su distribución por áreas (tabla 7.1) evidencia el éxito de la estrategia de unificación: la lógica de presentación se concentra en los componentes compartidos, mientras que las páginas (los wrappers) son notablemente ligeras y la capa de datos queda claramente separada. La clave, sin embargo, no es el número absoluto de líneas, sino su organización: antes de la refactorización cada par de páginas Coche/Moto exigía mantener dos copias casi idénticas; después, una única implementación sirve a ambas, de modo que la duplicación de componentes críticos queda erradicada y el coste de cualquier mejora se reduce a la mitad por construcción.

Tabla 7.1. Distribución aproximada de líneas de código por área del subsistema web.

Área del código	Líneas (aprox.)	Naturaleza
components/ (compartidos)	1540	Lógica de presentación reutilizable
features/ (wrappers+detalle)	1090	Envoltorios finos por disciplina
data/ (contenido)	720	Contenido editable (CMS estático)
styles + SCSS de componente	3750	Sistema de diseño y estilos

7.1.2. Caso de estudio: el fallo de color del CTA

Un incidente concreto durante el desarrollo ilustra de forma elocuente la importancia de respetar el sistema de tokens. El síntoma era que el verde de la llamada a la acción (CTA) final de la portada no coincidía con el verde de la barra de navegación, pese a que aparentemente ambos escribían la misma variable de color. La hipótesis inicial (un problema de caché) resultó falsa: se trataba de un fallo real. El diagnóstico reveló que, en `HomeSection.scss`, justo después de importar la paleta global con `@use`, unas variables locales «de respaldo» **sombreaban** a las oficiales, redefinían el verde principal con un tono distinto del corporativo y desviaban el amarillo; como esas redefiniciones tenían prioridad dentro del fichero, los verdes de la portada resolvían a un valor distinto del de la barra de navegación. La solución consistió en eliminar esas redefiniciones para que prevaleciera la paleta global, conservando únicamente la variable local `$dark-bg`, la única que no existe en el fichero de tokens. Tras el cambio, la verificación sobre el SCSS compilado confirmó que todos los verdes resolvían al mismo valor corporativo. La lección de arquitectura, incorporada como norma del proyecto, es clara: **no sombrear los tokens globales** dentro de los ficheros de componente; los colores deben provenir siempre del sistema de diseño centralizado.

7.1.3. Grado de cumplimiento de los objetivos del subsistema web

Confrontados los objetivos específicos del subsistema web (apartado 4.2) con el resultado, el balance es plenamente satisfactorio (tabla 7.2).

Tabla 7.2. Grado de cumplimiento de los objetivos del subsistema web.

Objetivo	Estado
Erradicar la duplicación de código	Cumplido: un componente compartido por funcionalidad
Desacoplar lógica y contenido	Cumplido: capa data/ como CMS estático
Elevar la UX/UI a un nivel profesional	Cumplido: sistema de marca y presentación cuidada
Coherencia de marca con tokens	Cumplido: tokens centralizados y norma de no sombreado
Reorganizar por dominios y alias	Cumplido: features/ y alias @
Manual para Marketing	Cumplido: README con la metáfora de los cajones
Diseño responsive en móvil	Cumplido: auditoría móvil superada

7.2. Resultados del subsistema de telemetría

Se presentan a continuación los resultados experimentales de la arquitectura dual Box/Edge, entrenada sobre tres circuitos rápidos (Monza, Spa y Silverstone) y evaluada sobre un circuito lento y revirado nunca visto (Mónaco).

7.2.1. Conjuntos de entrenamiento y test

Los datos proceden de la vuelta rápida de tres pilotos en cada circuito, extraída con fastf1; hay, por tanto, tres pilotos y tres vueltas por circuito. El conjunto consta de 7142 muestras: 5508 de entrenamiento (los tres circuitos rápidos) y 1634 de test aislado (Mónaco). La tabla 7.3 detalla la composición por circuito.

Tabla 7.3. Composición de los conjuntos de entrenamiento y test.

Circuito	Conjunto	Muestras	Vueltas	Pilotos
Monza	Entrenamiento	1822	3	3
Spa	Entrenamiento	1669	3	3
Silverstone	Entrenamiento	2017	3	3
Subtotal entrenamiento		5508	9	3
Mónaco	Test aislado	1634	3	3
Total		7142	12	3

La distribución de la variable objetivo (tabla 7.4) ilustra con claridad el cambio de dominio entre ambos conjuntos. En entrenamiento domina la octava marcha (1943 muestras, el 35,3 %) y la primera no aparece; en Mónaco la octava casi desaparece (21 muestras) y emergen la primera (92) y la segunda (444). Este desplazamiento justifica tanto el uso de

`class_weight="balanced"` como la magnitud de la caída de rendimiento entre entrenamiento y test que se analiza en la apartado 7.2.2.

Tabla 7.4. Distribución de la marcha engranada (nGear) por conjunto.

Marcha	Entrenamiento	Test (Mónaco)	Total
1	0	92	92
2	65	444	509
3	470	171	641
4	702	272	974
5	497	237	734
6	870	182	1052
7	961	215	1176
8	1943	21	1964
Total	5508	1634	7142

El filtro de rango físico (Sección 6.3.1) no descarta ninguna muestra, pues los datos de `fastf1` ya están dentro de rango. El único descarte se produce tras la ingeniería de características: la aceleración longitudinal `a_lon`, al calcularse por diferencia hacia atrás, deja indefinida la primera muestra de cada vuelta, que se elimina. El conjunto final de modelado queda así en 5499 muestras de entrenamiento y 1631 de test.

El análisis exploratorio del conjunto de entrenamiento se resume en la figura 7.1, que evidencia el fuerte desequilibrio de la variable objetivo, la estructura física de bandas por marcha en el plano velocidad–régimen y la alternancia recta/curva de una vuelta.

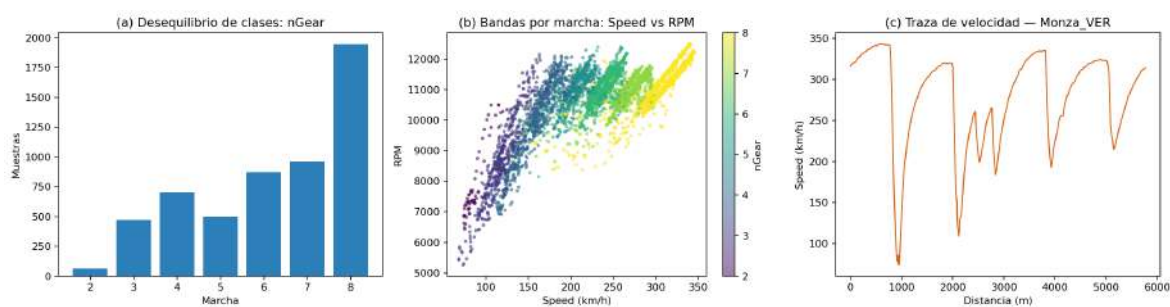


Figura 7.1. Análisis exploratorio del conjunto de entrenamiento: (a) desequilibrio de clases de la marcha; (b) bandas diagonales por marcha en el plano velocidad–RPM, evidencia de que el problema es aprendible; (c) traza de velocidad de una vuelta.

7.2.2. Generalización: del entrenamiento al test aislado en Mónaco

La validación se diseñó deliberadamente como un test de **dominio cruzado**: el modelo aprende exclusivamente de circuitos de alta velocidad y se evalúa sobre Mónaco, cuyo régimen

dinámico es estructuralmente distinto. Los estadísticos lo confirman: la velocidad media cae de 243,4 km/h en entrenamiento a 165,3 km/h en Mónaco (−78,2 km/h), el régimen medio de 10,693 a 9854 RPM y el acelerador medio del 79,0 % al 57,4 %. La consecuencia sobre la variable objetivo es aún más drástica: mientras el entrenamiento está dominado por la octava marcha (35,3 % de las muestras), en Mónaco esa marcha representa apenas el 1,3 % y el peso se desplaza hacia marchas bajas (27,2 % en segunda). No se trata, por tanto, de una partición aleatoria del mismo fenómeno, sino de un genuino domain shift que mide la capacidad real de extrapolación física del modelo.

Sobre este escenario, la degradación observada es notablemente contenida. La validación cruzada leave-one-circuit-out (honesta, sin fuga) sitúa al MLP del Box en 88,24 % $\pm$ 2,44 de accuracy y al árbol del Edge en 86,18 % $\pm$ 1,51, como muestra la figura 7.2. En el test final aislado de Mónaco, el MLP retiene un 78,05 % (F1 ponderada 0,757) y el árbol un 75,11 % (F1 0,733). La caída de aproximadamente diez puntos respecto a la validación cruzada no debe interpretarse como sobreajuste, sino como el coste esperable del salto de dominio: el modelo nunca ha visto la combinación de baja velocidad y marchas cortas característica del circuito urbano y, aun así, clasifica correctamente tres de cada cuatro muestras. La baja desviación entre particiones ($\pm$ 1,5 a 2,4 pp) refuerza esta lectura: el modelo es estable entre circuitos, signo de que ha capturado la **física del tren motriz**, la relación cinemática entre régimen de motor y velocidad de rueda, y no las particularidades memorísticas de un trazado concreto.

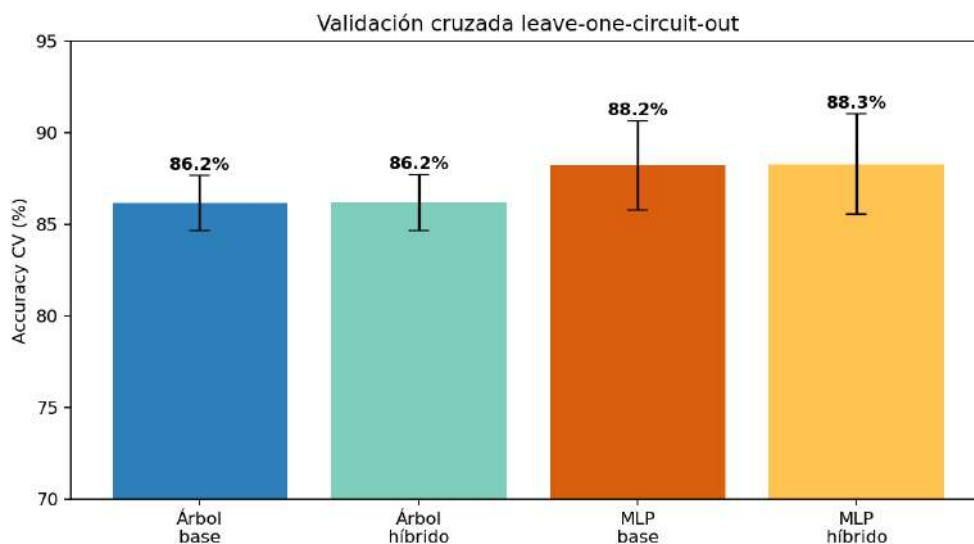


Figura 7.2. Validación cruzada leave-one-circuit-out. El MLP supera al árbol de forma consistente pero modesta; la variante híbrida no aporta mejora significativa.

Conviene subrayar el control metodológico que respalda estas cifras. El experimento de fuga de información (figura 6.2) cuantifica un sesgo optimista de 7,52 **puntos** cuando se emplea una partición barajada sobre una serie temporal autocorrelada, frente a la partición por circuitos. Reportar el 78 % de Mónaco, y no un 85 % a 90 % inflado por leakage, es lo que

confiere validez defendible a la conclusión. La figura 7.3 muestra dónde falla cada modelo: el error se concentra en marchas adyacentes, el tipo de error menos grave, lo que confirma que el modelo no confunde regímenes dinámicos lejanos.

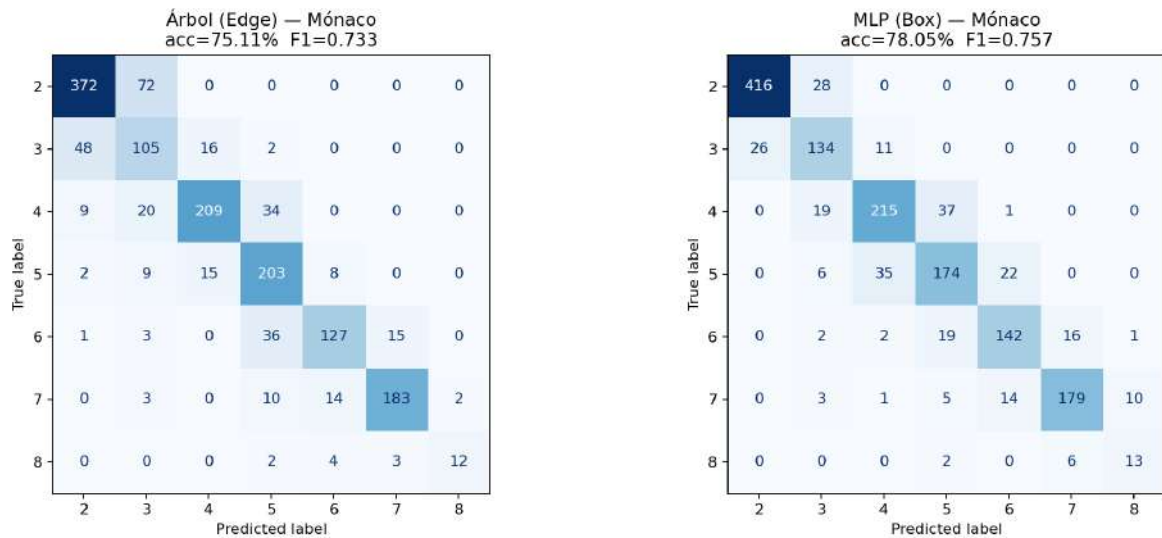


Figura 7.3. Matrices de confusión en el test aislado de Mónaco. Los errores se concentran en marchas adyacentes a la diagonal, el error menos grave.

7.2.3. Interpretación física del PCA

El análisis de componentes principales sobre el espacio de características causales concentra en torno al 90 % de la varianza en cuatro componentes, cada una con una lectura física directa, como ilustra la figura 7.4.

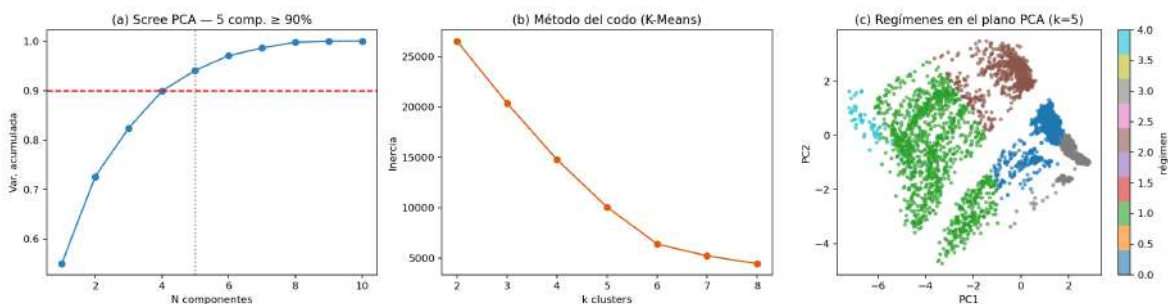


Figura 7.4. Detección de regímenes de conducción: (a) gráfico de sedimentación que identifica cuatro componentes para el 90 % de varianza; (b) método del codo para K-Means; (c) regímenes en el plano de las dos primeras componentes.

PC1 ($\approx 55\%$ de la varianza), eje de demanda de potencia. Las cargas dominantes son `power_proxy` (+0,39), `Throttle` (+0,38) y `Speed` (+0,37), todas con el mismo signo. Esta componente es el eje de entrega de energía longitudinal: separa las fases de máxima solicitud del tren motriz de las de baja demanda. De cara al monoplaza eléctrico, este eje gobierna el consumo instantáneo de la batería y la carga térmica del motor.

PC2 ($\approx 18\%$), eje transitorio. Pesan la aceleración longitudinal `a_lon` (+0,56) y `rpm_per_speed` (+0,41) en oposición a `Speed` (−0,35). Distingue las fases de frenada y reaceleración de la velocidad de crucero; captura dónde se usa la caja de cambios para gestionar el cambio de estado dinámico.

PC3 ($\approx 10\%$), eje de inercia. Dominado casi por completo por `coasting` (+0,94). Aísla las fases de rodadura sin acelerador ni freno, estado crítico en un EV porque es donde la frenada regenerativa decide cuánta energía recuperar.

PC4 ($\approx 8\%$), eje aerodinámico. Cargado por `drs_active` (+0,90). Representa la configuración de baja resistencia en recta como un modo de operación físicamente distinguible.

La lectura conjunta es elocuente: las cuatro componentes reconstruyen, sin supervisión, los cuatro grandes modos operativos de un monoplaza, demanda de potencia, transitorios, rodadura inercial y configuración aerodinámica. Esto valida que el espacio de características no es redundante ni arbitrario, sino que codifica la estructura física del problema.

7.2.4. Rendimiento de modelos: Árbol de Decisión frente a MLP

La comparación confirma la hipótesis de partida con un matiz cuantitativo importante. El MLP del Box supera al árbol del Edge de forma consistente pero modesta: +2,1 pp en validación cruzada honesta (88,24 % frente a 86,18 %) y +2,9 pp en el test de Mónaco (78,05 % frente a 75,11 %). La ventaja del MLP es real pero acotada, y se concentra en la frontera de decisión.

La clave la da la importancia de características del árbol (figura 7.5): `rpm_per_speed` concentra alrededor del 0,83 de la importancia total y `a_lon` aporta el resto, quedando las demás variables en valores marginales. Esto es físicamente esperable y casi tautológico: la marcha engranada es, por construcción de la transmisión, una función discreta de la relación entre régimen de motor y velocidad de rueda. El modelo no descubre una correlación espuria, sino que **reconstruye las relaciones de la caja de cambios**, como se aprecia en las primeras ramas del árbol (figura 7.6). De ahí que la reducción a solo `Speed` y `RPM` provoque una caída muy contenida: +6,3 pp en el árbol y +3,9 pp en el MLP. Con dos variables el problema queda casi determinado, lo que abre la puerta a un Edge mínimo de dos sensores.

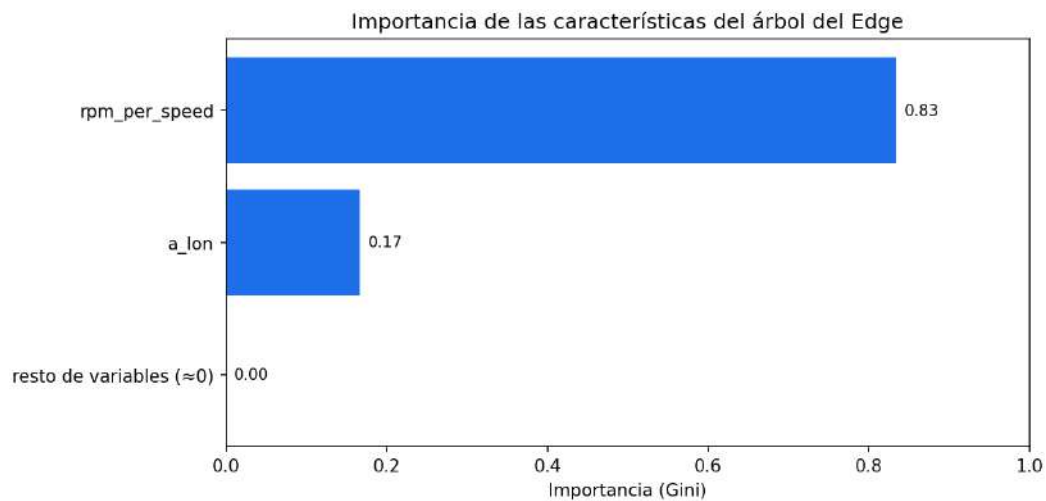


Figura 7.5. Importancia de las características del árbol del Edge. El cociente rpm_per_speed domina, reconstruyendo la relación física de la caja de cambios.

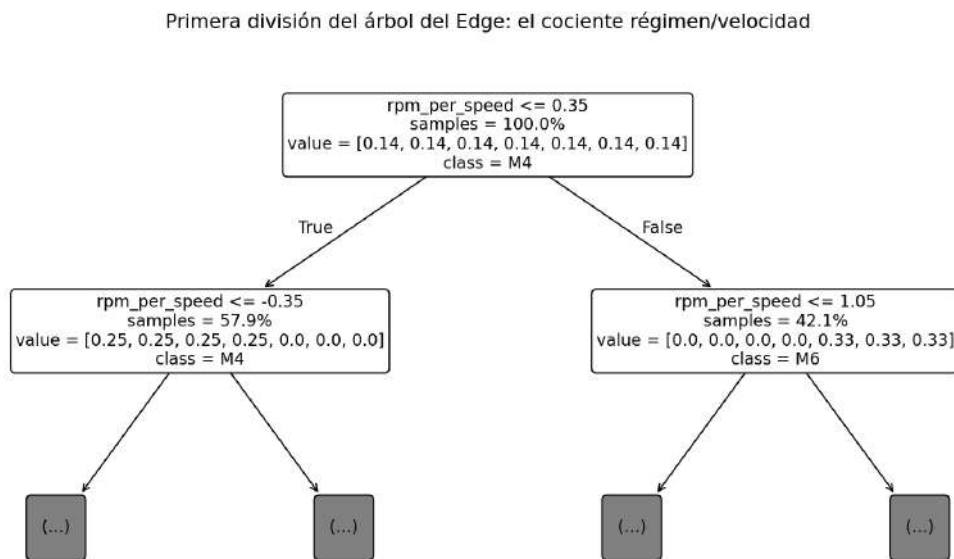


Figura 7.6. Primera división del árbol del Edge. La decisión raíz es un umbral sobre el cociente régimen/velocidad, legible y físicamente interpretable.

El árbol completo del Edge tiene una profundidad de 5 y solo 9 hojas, lo que confirma su sencillez y su carácter interpretable. La figura 7.7 lo reproduce entero: salvo dos cortes auxiliares de importancia casi nula, todas las decisiones se toman sobre el cociente régimen/velocidad y, en segundo plano, sobre la aceleración longitudinal, las dos variables que concentran la importancia del modelo. Por su tamaño, la figura se aprecia mejor ampliándola en la versión digital del documento.

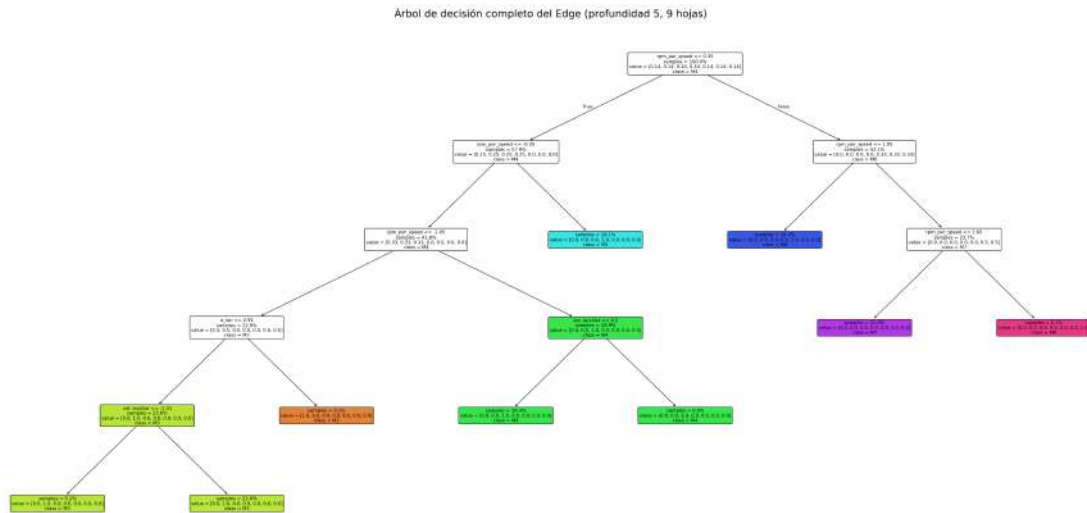


Figura 7.7. Árbol de decisión completo del Edge (profundidad 5, 9 hojas). Cada nodo muestra el umbral de decisión, el porcentaje de muestras y la clase mayoritaria. El modelo decide casi por completo a partir del cociente régimen/velocidad (`rpm_per_speed`) y, en menor medida, de la aceleración longitudinal (`a_lon`).

Que el MLP degrade menos al reducir variables (+3,9 frente a +6,3 pp) revela dónde aporta valor la red. El árbol, con fronteras ortogonales a los ejes, aproxima de forma escalonada las zonas de cambio de marcha, los puntos donde dos marchas comparten un rango de `rpm_per_speed` por histéresis del cambio. El MLP, gracias a sus fronteras no lineales y suaves, resuelve mejor esas transiciones, que son precisamente las muestras ambiguas donde se concentra el error. La variante híbrida (incorporación de regímenes vía one-hot de K-Means) aporta diferencias inferiores a 0,4 pp, dentro del ruido entre particiones: no se justifica su coste de complejidad, conclusión metodológicamente valiosa de cara a la parsimonia del despliegue.

7.2.5. Viabilidad de la arquitectura Edge: latencia y huella

Los tiempos de cómputo sostienen cuantitativamente la división Box/Edge (figura 7.8). El árbol desplegable en la Raspberry Pi entrena en 0,081 s, infiere a 1,76 μ s por muestra en lote y a 1,90 ms por muestra en el escenario realista de inferencia muestra a muestra, y ocupa 31,7 kB. El MLP del Box entrena en 4,167 s (≈ 51 veces más), infiere a 2,23 ms en modo individual y pesa 328,1 kB (≈ 10 veces más). El árbol es, por tanto, un orden de magnitud más ligero en memoria y comparativamente más rápido en la inferencia individual, que es la que importa en el volante.

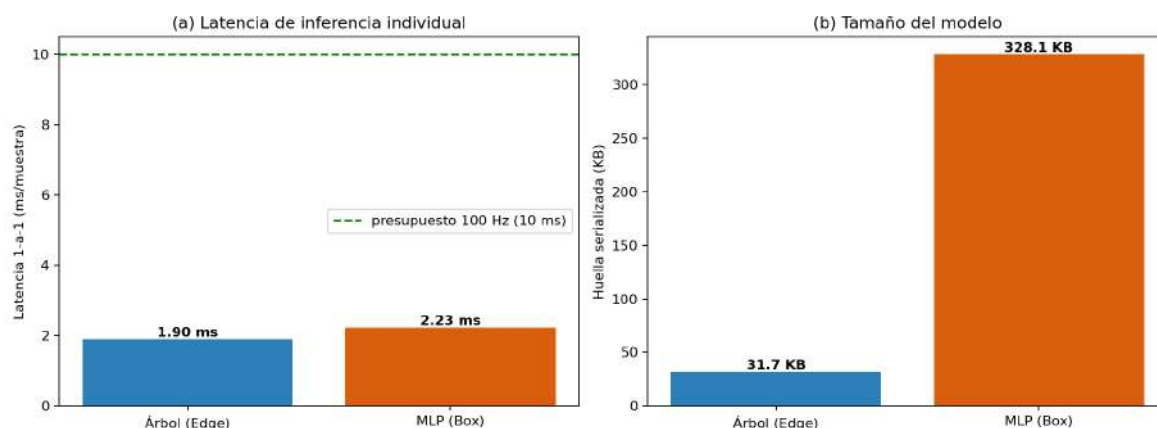


Figura 7.8. Viabilidad del Edge: (a) latencia de inferencia individual frente al presupuesto de 10 ms del lazo a 100 Hz; (b) huella de memoria serializada. El árbol cumple ambos requisitos con amplio margen.

La pregunta crítica es si 1,90 ms de latencia bastan para decidir en tiempo real. La respuesta es afirmativa, con dos matizaciones. Primera, el contexto físico: a 300 km/h (≈ 83 m/s) el coche recorre unos 16 cm en 1,90 ms. Para lazos de control de gestión térmica o de mezcla de frenada regenerativa, que operan típicamente a 100 Hz (presupuesto de 10 ms por ciclo), una latencia inferior a 2 ms deja holgura más que suficiente. Segunda, y decisiva para la defensa: ese valor es un **límite superior conservador**, dominado por la sobrecarga del intérprete de Python y de scikit-learn, no por la aritmética del árbol, que en lote se mide en 1,76 μ s, mil veces menos. Un despliegue en producción sobre la Raspberry Pi, con el árbol transpilado a C o servido mediante un runtime ligero, reduciría la latencia efectiva a la escala de los microsegundos.

La conclusión arquitectónica es nítida. El árbol del Edge no solo cabe holgadamente en la memoria de la Raspberry Pi, sino que cumple los requisitos de tiempo real con un margen que tolera incluso el sobrecoste de un entorno interpretado. El MLP del Box, más preciso pero diez veces más pesado y sin restricción de latencia al operar tras la sesión, queda correctamente asignado al análisis de alta capacidad en garaje. La penalización de exactitud por desplegar el árbol en lugar del MLP (en torno a 3 pp) es un precio razonable a cambio de la autonomía, la frugalidad energética y el determinismo temporal que exige la inferencia embarcada. La arquitectura dual queda así validada por evidencia cuantitativa de precisión, latencia y huella de memoria.

7.2.6. Valoración económica: el valor de la herramienta para el equipo

Más allá del coste material prácticamente nulo del proyecto (Sección 4.5), lo relevante es el valor que aporta al equipo disponer de esta herramienta. El sistema entrega a ISC Racing Team una capacidad de la que antes carecía: convertir la telemetría de su monopla en información útil en tiempo real, sobre la Raspberry Pi de bajo coste que el equipo ya posee

y sin ningún gasto recurrente. Esa capacidad se traduce en ventaja en pista: un estimador embarcado de la marcha y del régimen de conducción es la base sobre la que apoyar decisiones de gestión energética, como la mezcla de frenada regenerativa o la gestión térmica, y queda inmediatamente disponible para el coche, en lugar de depender de un análisis manual posterior a la sesión. A esto se añade el valor operativo del análisis del Box, que permite revisar cada tanda con mayor profundidad. En conjunto, el equipo obtiene un activo técnico reutilizable y ampliable a coste cero, lo que libera presupuesto para la parte mecánica y electrónica del vehículo.

El análisis de viabilidad confirma, además, que el coste operativo del sistema embarcado es mínimo y sin componente recurrente. El modelo del Edge ocupa 31,7 kB y, en su variante experimental de dos variables, tan solo 12,8 kB, cifras despreciables frente a los recursos de la Raspberry Pi, lo que descarta la necesidad de hardware adicional. Su frugalidad de sensores (dependencia de únicamente dos señales robustas del bus en la variante mínima) reduce la superficie de fallo y, con ello, los costes de mantenimiento y de indisponibilidad. La ausencia de licencias, de infraestructura de servidor y de costes de comunicación remota (consecuencia directa del paradigma de Edge Computing) consolida un modelo de coste total de propiedad prácticamente nulo tras la inversión inicial en horas de ingeniería, lo que hace el sistema especialmente adecuado para un equipo universitario.

La comparativa entre el modelo híbrido de referencia (diez características derivadas) y el experimental minimalista (dos señales crudas) refuerza esta conclusión desde la perspectiva de la ingeniería de despliegue. Aunque el experimental sacrifica 6,1 puntos de exactitud (68,97 % frente a 75,11 %), cumple la restricción de 100 Hz con holgura, elimina toda la cadena de preprocesamiento con estado, reduciendo la latencia real del sistema de extremo a extremo y su no-determinismo, y resulta directamente auditable y transpilable a código embebido. Esta asignación, el modelo simple y robusto en el Edge, el modelo completo y preciso en el Box, materializa la arquitectura dual que sostiene el trabajo.

Capítulo 8

Conclusiones y Trabajos Futuros

8.1. Conclusiones

El trabajo ha construido la plataforma de datos de extremo a extremo de ISC Racing Team a través de sus dos subsistemas, y permite extraer conclusiones diferenciadas para cada uno y una conclusión integradora, todas alineadas con los objetivos formulados en el Capítulo 4.

8.1.1. Sobre la plataforma web

El subsistema web ha llevado la plataforma del equipo desde una base duplicada y acoplada hasta una arquitectura limpia, coherente y mantenible, sin renunciar a una presentación visual cuidada. En cifras, la plataforma cubre las **24 rutas** de las dos disciplinas con un único componente compartido por funcionalidad y unas 7200 líneas de código organizadas de modo que la lógica de presentación (≈ 1540 líneas) sirve a ambas divisiones, mientras que los *wrappers* por disciplina apenas suman una docena de líneas cada uno. La conclusión principal es que un análisis arquitectónico riguroso (la observación de que dos disciplinas con la misma estructura solo difieren en el contenido) se traduce en una solución que **erradica la duplicación** y reduce a la mitad el coste de cada mejora, pues se implementa una sola vez. Más allá del código, el proyecto **devuelve al equipo la autonomía** para mantener su propia web, al desacoplar la lógica del contenido y documentarlo con un manual accesible. El caso del fallo de color reforzó, además, el valor práctico de un sistema de diseño centralizado y la norma de no sombrear sus tokens.

8.1.2. Sobre el subsistema de inteligencia de telemetría

El subsistema de telemetría, con su arquitectura dual de modelos, permite extraer cuatro conclusiones principales.

La primera es que la **arquitectura dual Box/Edge queda justificada por evidencia cuantitativa**, y no por presunción de diseño. El MLP del Box supera al árbol del Edge de

forma consistente pero modesta, +2,1 puntos en validación cruzada honesta y +2,9 puntos en el test de Mónaco, mientras que el árbol es un orden de magnitud más ligero (31,7 kB frente a 328,1 kB) y resuelve la inferencia individual con menor latencia. La penalización de exactitud por embarcar el árbol en lugar del MLP, en torno a 3 puntos, es un precio razonable a cambio de la autonomía, el determinismo temporal y la frugalidad que exige el volante. Asignar cada modelo al entorno donde sus propiedades son óptimas es, por tanto, la decisión de ingeniería correcta, y está respaldada por medidas de precisión, latencia y huella de memoria.

La segunda es que el **rigor metodológico se ha demostrado, no afirmado**. El experimento controlado de fuga de información cuantifica un sesgo optimista de 7,52 puntos cuando se emplea una partición barajada sobre una serie temporal autocorrelada en lugar de la partición por circuitos. Reportar el 78,05 % honesto de Mónaco, y no una cifra inflada por data leakage, es lo que confiere validez defendible al resultado. La ingeniería de características íntegramente causal, verificada mediante la propiedad de invariancia de prefijo, garantiza además que el modelo del Edge es desplegable tal cual sobre la Raspberry Pi, sin variables que dependan de información futura.

La tercera es que el sistema **generaliza de forma honesta a un dominio nunca visto**. Entrenado solo en circuitos rápidos y evaluado sobre el régimen lento y revirado de Mónaco, un genuino domain shift, no una partición aleatoria del mismo fenómeno, el modelo retiene tres de cada cuatro clasificaciones correctas, con los errores concentrados en marchas adyacentes y una baja desviación entre circuitos. Esto evidencia que el modelo ha capturado la física del tren motriz, la relación cinemática entre régimen de motor y velocidad de rueda, reconstruida sin supervisión por el PCA y dominante en la importancia de características del árbol, y no las particularidades memorísticas de un trazado.

La cuarta es que la metodología es **trasladable al monoplaza eléctrico de ISC Racing Team**. La física longitudinal sobre la que se ha validado el pipeline es común a cualquier vehículo de competición, y el resultado de que dos señales crudas (velocidad y régimen) casi determinan el problema abre la puerta a un Edge mínimo, robusto y de muy bajo coste. El sistema constituye, en suma, una solución de extremo a extremo, adquisición, modelado honesto y despliegue embarcado viable, concebida para una necesidad real del equipo y con un coste material prácticamente nulo.

8.1.3. Conclusión integradora: el dominio del dato de extremo a extremo

Tomados en conjunto, ambos subsistemas demuestran la tesis central del trabajo: que la competitividad de un equipo de Formula Student pasa por el **dominio de su propio dato a lo largo de todo su ciclo de vida**, y que ese dominio puede lograrse con recursos modestos si se aplica una misma disciplina de ingeniería en cada eslabón. La plataforma web y el motor de telemetría no son dos proyectos contiguos, sino las dos caras de una misma estrategia: comparten la filosofía de separar el dato de la lógica. En la web esa separación

es la capa *data*/; en la telemetría, la ingeniería de características causales y los parámetros congelados del pipeline. Comparten también la apuesta por la frugalidad y la autonomía: la web no necesita servidor ni licencias, y el modelo embarcado infiere sin conexión ni hardware costoso. Y los dos se orientan a un usuario real del equipo. La plataforma capta y presenta el dato que financia y nutre al equipo; el motor lo explota para mejorar el coche; y la convergencia natural de ambos, un panel de visualización de la telemetría y de las predicciones embarcadas dentro de la propia web, es la línea de trabajo que cierra el ciclo de extremo a extremo.

8.2. Limitaciones del trabajo

Conviene delimitar con honestidad el alcance de los resultados y las restricciones del estudio en su estado actual.

En el **subsistema de telemetría**, la principal limitación es que la validación se ha realizado sobre telemetría pública de Fórmula 1, no sobre el monoplaza eléctrico del equipo. Aunque la física longitudinal es común a ambos vehículos, la transferencia de las cifras exactas al coche de ISC requerirá recalibrar los filtros y umbrales y verificar los resultados con datos propios. El conjunto de datos es, además, **reducido** (7142 muestras, tres pilotos y una vuelta rápida por circuito): suficiente como banco de pruebas metodológico, pero no para extraer conclusiones definitivas sobre la capacidad máxima de los modelos. El problema modelado (la marcha engranada) se eligió por ser una **función física bien definida** que actúa como termómetro de la generalización; no es, en sí mismo, la variable de mayor valor operativo, sino la prueba de que el *pipeline* es honesto y trasladable a objetivos más útiles (consumo, temperatura, par regenerado). Las cifras de latencia, por último, se han medido en un entorno interpretado (Python/scikit-learn) y constituyen un límite superior; la latencia real embarcada, tras la transpilación a código nativo, queda pendiente de medición sobre la propia Raspberry Pi.

En el **subsistema web**, el alcance se limita al *frontend*: el sistema de gestión de contenido es estático y versionado, sin *backend* ni base de datos, lo que es adecuado para el caso de uso actual pero condiciona funciones que requieren autenticación o edición en línea (de ahí el trabajo futuro del Apartado 8.3.1). La validación de la plataforma ha sido funcional y de coherencia de marca; no se ha realizado todavía una auditoría formal de accesibilidad ni un estudio cuantitativo de experiencia de usuario con usuarios reales.

8.3. Trabajos futuros

8.3.1. Líneas del subsistema web

La plataforma está en producción con todos sus textos, fotografías, vídeos y documentos actualizados y propios de cada disciplina. Sobre esa base ya consolidada, las líneas de evolución natural son: incorporar internacionalización (español e inglés) para reforzar la comunicación con patrocinadores internacionales; optimizar el posicionamiento en buscadores mediante renderizado en servidor o prerrenderizado estático si el tráfico orgánico cobrara relevancia; y realizar una auditoría de accesibilidad que garantice el cumplimiento de los estándares web.

La línea más ambiciosa, ya planteada por la autora en su nueva responsabilidad como **jefa de digitalización de ISC Racing Team**, es la **construcción de un backend con autenticación**: dotar a la plataforma de un sistema de acceso mediante usuario y contraseña que permita a los miembros del equipo entrar a un área privada. Este paso supone evolucionar desde el modelo actual de contenido estático hacia una arquitectura con servidor, base de datos y gestión de identidades y roles, y habilitaría funciones reservadas, edición de contenido desde el propio navegador sin tocar el repositorio, paneles internos por departamento y, de forma natural, el panel de visualización de telemetría descrito más abajo. El diseño desacoplado del frontend actual (capa de datos separada de la lógica de presentación) facilita esta transición, pues permite sustituir la fuente de datos estática por una API autenticada sin reescribir los componentes de presentación.

8.3.2. Líneas del subsistema de telemetría

La continuación natural del trabajo es **sustituir el banco de pruebas de Fórmula 1 por la telemetría real del monoplace eléctrico** de ISC Racing Team. El pipeline se ha construido deliberadamente para que esa transición tenga un coste de adaptación reducido, pero requerirá recalibrar los filtros de rango físico y los umbrales congelados a las características del vehículo propio, así como construir un histórico de sesiones suficientemente denso.

En segundo lugar, conviene **incorporar variables específicas del tren motriz eléctrico** que la telemetría de combustión no ofrece: la temperatura y el estado de carga de la batería, el par regenerado en la frenada y la carga térmica del motor eléctrico. El análisis PCA ya ha identificado los ejes físicos donde estas variables resultan determinantes, en particular el eje de inercia, asociado a la rodadura sin acelerador, que es donde la frenada regenerativa decide cuánta energía recuperar, lo que sugiere que su inclusión enriquecería la capacidad del sistema para optimizar la gestión energética.

En tercer lugar, procede **medir la latencia real sobre la Raspberry Pi** en condiciones de despliegue, transpilando el árbol a código nativo (C) o sirviéndolo mediante un runtime ligero para eliminar la sobrecarga del intérprete de Python. Las cifras de latencia reportadas

en este trabajo son un límite superior conservador medido en un entorno interpretado; cabe esperar que el despliegue embarcado las reduzca a la escala de los microsegundos, lo que habilitaría su uso en lazos de control más exigentes que los de gestión térmica o de mezcla de frenada regenerativa. Finalmente, una vez disponible un histórico suficiente de telemetría del propio monoplaza, procede cerrar el ciclo de extremo a extremo, adquisición en pista, inferencia embarcada y realimentación al sistema de control, validando sobre el vehículo real las cifras de latencia y de generalización obtenidas en este trabajo sobre el banco de pruebas.

8.3.3. Línea integradora: el panel de telemetría en la plataforma

La línea de mayor valor estratégico es la **convergencia de ambos subsistemas**: incorporar a la plataforma web un módulo de visualización que consuma las predicciones del motor de telemetría, marcha estimada, regímenes de conducción detectados por el PCA, indicadores de gestión energética, y los presente al equipo técnico y, en su versión divulgativa, a los patrocinadores. La capa de datos versionada de la web (apartado 5.2.2) y el formato serializado y frugal de los artefactos de inferencia (apartado 6.3.3) hacen que esta integración sea arquitectónicamente natural, y materializaría de forma tangible la plataforma de datos de extremo a extremo que vertebra este trabajo.

Bibliografía

- [1] FastF1 contributors, *FastF1: Python package for accessing F1 telemetry data*, <https://docs.fastf1.dev/>, Consultado en 2026, 2023.
- [2] F. Pedregosa, G. Varoquaux, A. Gramfort et al., «Scikit-learn: Machine Learning in Python», *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.
- [3] Meta Open Source. «React: The library for web and native user interfaces», visitado jun. de 2026. dirección: <https://react.dev/>
- [4] Vite contributors. «Vite: Next Generation Frontend Tooling», visitado jun. de 2026. dirección: <https://vite.dev/>
- [5] Robert Bosch GmbH, *CAN Specification, Version 2.0*, Robert Bosch GmbH, Stuttgart, Alemania, 1991.
- [6] W. Shi, J. Cao, Q. Zhang, Y. Li y L. Xu, «Edge Computing: Vision and Challenges», *IEEE Internet of Things Journal*, vol. 3, n.º 5, págs. 637-646, 2016. doi: 10.1109/JIOT.2016.2579198
- [7] AMZ Racing, ETH Zürich. «AMZ Racing — Formula Student Team of ETH Zurich», visitado jun. de 2026. dirección: <https://www.amzracing.ch/>
- [8] Formula Student Team Delft. «Formula Student Team Delft — TU Delft», visitado jun. de 2026. dirección: <https://www.fsteamdelft.nl/>
- [9] S. Kaufman, S. Rosset, C. Perlich y O. Stitelman, «Leakage in Data Mining: Formulation, Detection, and Avoidance», *ACM Transactions on Knowledge Discovery from Data*, vol. 6, n.º 4, págs. 1-21, 2012. doi: 10.1145/2382577.2382579

Apéndice A

Alineación del proyecto con los Objetivos de Desarrollo Sostenible

El proyecto contribuye, de forma directa o instrumental, a cuatro de los Objetivos de Desarrollo Sostenible de la Agenda 2030 de Naciones Unidas.

ODS 7, Energía asequible y no contaminante

El pipeline analítico se concibe como soporte a la transición del equipo hacia un powertrain eléctrico. El análisis de componentes principales identificó explícitamente los ejes físicos asociados a la demanda de potencia y a la rodadura inercial, que son los que gobiernan el consumo instantáneo de la batería y la oportunidad de recuperación energética. Una explotación inteligente de la telemetría orientada a estos ejes habilita una gestión más eficiente de la energía a bordo, en línea con la promoción de una energía limpia y eficiente.

ODS 9, Industria, innovación e infraestructura

El trabajo materializa una innovación tecnológica concreta en sus dos subsistemas: una plataforma web mantenible y reutilizable que dota al equipo de una infraestructura digital sostenible y autónoma, activo clave para su continuidad y su capacidad de captar recursos, y una arquitectura dual Box/Edge con inferencia embarcada en hardware de bajo coste (Raspberry Pi), construida sobre una metodología de ingeniería de datos rigurosa y reproducible. El paradigma de Edge Computing y la transpilación de modelos ligeros a hardware limitado constituyen una infraestructura de I+D resiliente y de bajo coste, accesible para un equipo universitario y transferible a otros contextos de instrumentación de vehículos.

ODS 12, Producción y consumo responsables

La frugalidad es un principio rector del sistema: el modelo del Edge depende de muy pocas señales y de una huella de memoria mínima, lo que reduce la superficie de fallo, el mantenimiento y la necesidad de hardware adicional. El análisis de la fase de rodadura inercial sienta las bases para optimizar la frenada regenerativa, esto es, para recuperar energía que de otro modo se disiparía, favoreciendo un consumo de recursos más responsable.

ODS 13, Acción por el clima

Al sustentar la transición del monoplaza hacia la movilidad eléctrica y al hacerlo con una arquitectura de coste material y energético prácticamente nulo, el proyecto se alinea con la acción por el clima. La apuesta por la electrificación de la competición y por una explotación eficiente de la energía contribuye, aunque sea a la escala de un equipo universitario, a la difusión de tecnologías de movilidad sostenible.

Apéndice B

Material complementario

B.1. Disponibilidad del código fuente

El código completo de los dos subsistemas está disponible en los siguientes repositorios públicos de GitHub:

- Plataforma web (frontend React/Vite):
<https://github.com/gabrieladorre-design/isc-web>
- Sistema de inteligencia de telemetría (pipeline de Machine Learning):
<https://github.com/gabrieladorre-design/isc-telemetry>

B.2. Fragmentos de código relevantes

Este anexo no pretende documentar la implementación de forma exhaustiva, sino reproducir, de forma puntual, los dos fragmentos de código que mejor ilustran las **decisiones de diseño** centrales del trabajo, discutidas en los Capítulos 6 y 7. El primero materializa la ingeniería de características **causal**: nótese el uso de derivada hacia atrás y media móvil sin centrar para `a_lon`, y el umbral de `corner_zone` congelado del entrenamiento y reaplicado idéntico en Mónaco, garantías de causalidad y de ausencia de fuga de información que hacen el modelo desplegable en el Edge tal cual.

```
1 def engineer_causal(df: pd.DataFrame, corner_speed_kmh: float | None =  
  None):  
2     df = df.copy()  
3  
4     def _per_lap(g: pd.DataFrame) -> pd.DataFrame:  
5         g = g.copy()  
6         g["Speed_ms"] = g["Speed"] / 3.6  
7         dt = g["Time_s"].diff().replace(0, np.nan)  
8         a = g["Speed_ms"].diff() / dt                                # diff hacia atras  
    = causal
```

```

9         g["a_lon"] = a.rolling(window=5, min_periods=1).mean() # SIN
center
10         return g
11
12     df = df.groupby("lap_id", group_keys=False).apply(_per_lap)
13     df["power_proxy"] = df["RPM"] * df["Throttle"] / 100.0
14     df["drs_active"] = df["DRS"].isin([10, 12, 14]).astype(float)
15     df["coasting"] = ((df["Throttle"] < 5.0) & (df["Brake"] < 0.5)).
astype(float)
16     df["rpm_per_speed"] = df["RPM"] / (df["Speed"] + 1e-3)
17     if corner_speed_kmh is None: # umbral congelado
del TRAIN
18         corner_speed_kmh = float(df["Speed"].quantile(0.33))
19     df["corner_zone"] = (df["Speed"] < corner_speed_kmh).astype(float)
20     df["accel_phase"] = ((df["Throttle"] > 80.0) & (df["a_lon"] > 0)).
astype(float)
21     return df, corner_speed_kmh

```

Listado B.1. Ingeniería de características causal, desplegable en el Edge.

El segundo ilustra la decisión metodológica que sostiene la honestidad de las métricas: la validación cruzada por circuitos (leave-one-circuit-out) mediante GroupKFold, con una factory que construye una instancia limpia del Pipeline por partición (evitando todo estado heredado) y que reajusta el escalado y el detector de régimen solo con los datos de entrenamiento de cada partición.

```

1 def grouped_cv(factory, X, y, groups, use_sample_weight: bool) -> dict:
2     n = groups.nunique()
3     gkf = GroupKFold(n_splits=n)
4     accs, f1s = [], []
5     for fold, (tr, va) in enumerate(gkf.split(X, y, groups), start=1):
6         X_tr, X_va = X.iloc[tr], X.iloc[va]
7         y_tr, y_va = y.iloc[tr], y.iloc[va]
8         pipe = factory() # instancia limpia
por fold
9         if use_sample_weight:
10             fit_with_weights(pipe, X_tr, y_tr)
11         else:
12             pipe.fit(X_tr, y_tr)
13             yh = pipe.predict(X_va)
14             accs.append(accuracy_score(y_va, yh))
15             f1s.append(f1_score(y_va, yh, average="weighted"))
16     return {"acc_mean": float(np.mean(accs)), "acc_std": float(np.std(
accs)),
17             "f1_mean": float(np.mean(f1s)), "f1_std": float(np.std(f1s
))}

```

Listado B.2. Validación cruzada agrupada por circuito.

El resto de la implementación, la adquisición desde la fuente local de datos, la limpieza por vuelta y la auditoría de latencia, se describe a nivel de decisión de diseño en el Capítulo 6, sin reproducir aquí su código por no aportar valor adicional a la comprensión del trabajo.

B.3. Tablas extensas

La tabla B.1 recoge el detalle de la validación cruzada honesta y del test aislado de Mónaco para los dos modelos en su configuración base. La variante híbrida (con detector de régimen) difiere en menos de 0,4 puntos, dentro del ruido entre particiones, lo que justifica descartar la complejidad añadida del detector de régimen en el despliegue.

Tabla B.1. Resultados de la validación cruzada leave-one-circuit-out y del test aislado de Mónaco. Exactitud en % y F1 ponderada.

Modelo	CV exact. (%)	CV F1	Mónaco exact. (%)	Mónaco F1
Árbol (Edge)	86,18 ± 1,51	0,860 ± 0,020	75,11	0,733
MLP (Box)	88,24 ± 2,44	0,879 ± 0,029	78,05	0,757

La tabla B.2 sintetiza las métricas de viabilidad embarcada que sustentan la asignación de cada modelo a su entorno.

Tabla B.2. Métricas de viabilidad del Edge frente al Box.

Modelo	Latencia individual	Huella	Entrenamiento
Árbol Edge	1,90 ms	31,7 kB	0,081 s
Árbol Edge (2 vars.)	< 2 ms	12,8 kB	< 0,1 s
MLP Box	2,23 ms	328,1 kB	4,167 s

B.4. Fragmentos de código del subsistema web

Esta sección reproduce los listados que mejor ilustran las decisiones de arquitectura del subsistema web discutidas en el Capítulo 5. El Listado B.3 muestra la normalización de rutas mediante el alias @, declarado de forma complementaria en vite.config.js (para el empaquetador) y en jsconfig.json (para el editor).

```

1 // vite.config.js -- alias @ -> ./src en tiempo de compilacion
2 import { defineConfig } from 'vite'
3 import react from '@vitejs/plugin-react'
4 import { fileURLToPath, URL } from 'node:url'
5 export default defineConfig({
6   plugins: [react()],

```

```

7   resolve: {
8     alias: { '@': fileURLToPath(new URL('./src', import.meta.url)) },
9   },
10  })
11
12 // jsconfig.json -- alias @ para el editor (autocompletado)
13 {
14   "compilerOptions": { "baseUrl": ".", "paths": { "@/*": ["src/*"] } },
15   "include": ["src"]
16 }

```

Listado B.3. Configuración del alias @ en Vite y en el editor.

El Listado B.4 ilustra el patrón **componente compartido + wrapper fino**, núcleo de la erradicación de la duplicación (apartado 5.2.1): un único componente de presentación, agnóstico de la disciplina, recibe por props el contenido que cada wrapper le inyecta desde la capa data/.

```

1 // components/data-display/Garage.jsx -- componente compartido (agnostico
  )
2 export default function Garage({ title, vehicles, basePath }) {
3   return (
4     <section className="garage">
5       <h1>{title}</h1>
6       <div className="garage__grid">
7         {vehicles.map(v => (
8           <VehicleCard key={v.id} vehicle={v} to={`/${basePath}/${v.id}`}
9         </div>
10       </div>
11     </section>
12   )
13 }
14
15 // features/formula/GaragePage.jsx -- wrapper fino (Coche)
16 import Garage from '@components/data-display/Garage'
17 import { coches } from '@data/formula/coches'
18 export default function GaragePage() {
19   return <Garage title="Garaje" vehicles={coches} basePath="/garaje" />
20 }
21
22 // features/moto/GaragePage.jsx -- wrapper fino (Moto)
23 import Garage from '@components/data-display/Garage'
24 import { motos } from '@data/moto/motos'
25 export default function GaragePageMoto() {
26   return <Garage title="Garaje" vehicles={motos} basePath="/moto/garaje"
27   />

```

```
27 }
```

Listado B.4. Patrón componente compartido alimentado por wrappers finos.

El Listado B.5 recoge el contador numérico animado, ejemplo del uso eficiente de la API `IntersectionObserver` para disparar la animación solo cuando el elemento entra en el área visible, evitando escuchar de forma continua el evento de scroll.

```
1 // hook que activa una animacion al entrar el elemento en pantalla
2 import { useEffect, useRef, useState } from 'react'
3 export function useOnScreen(threshold = 0.3) {
4   const ref = useRef(null)
5   const [visible, setVisible] = useState(false)
6   useEffect(() => {
7     const obs = new IntersectionObserver(
8       ([entry]) => { if (entry.isIntersecting) setVisible(true) },
9       { threshold }
10    )
11    if (ref.current) obs.observe(ref.current)
12    return () => obs.disconnect() // limpieza al desmontar
13  }, [threshold])
14  return [ref, visible]
15 }
```

Listado B.5. Animación de aparición eficiente con `IntersectionObserver`.