UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE SOFTWARE

Tema:

SISTEMA WEB CON MODELO PREDICTIVO DE DESERCIÓN PARA EL GIMNASIO STRONG 2.0

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero de Software

ÁREA: Desarrollo de software

LÍNEA DE INVESTIGACIÓN: Software, Tecnologías de la información y Ciencia de datos

AUTOR: Erick Daniel Fuelpaz Valverde

TUTOR: Ing. Leonardo David Torres Valverde, Mg

Ambato – Ecuador

julio - 2026

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA WEB CON MODELO PREDICTIVO DE DESERCIÓN PARA EL GIMNASIO STRONG 2.0, desarrollado bajo la modalidad Proyecto de Investigación por el señor Erick Daniel Fuelpaz Valverde, estudiante de la Carrera de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, julio de 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
LEONARDO DAVID
TORRES VALVERDE

Ing. Leonardo David Torres Valverde, Mg

Tutor

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA WEB CON MODELO PREDICTIVO DE DESERCIÓN PARA EL GIMNASIO STRONG 2.0 es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, julio 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
ERICK DANIEL
FUELPAZ VALVERDE

Erick Daniel Fuelpaz Valverde

C.C. 1850178870

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, julio 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
ERICK DANIEL
FUELPAZ VALVERDE

Erick Daniel Fuelpaz Valverde

C.C. 1850178870

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Erick Daniel Fuelpaz Valverde, estudiante de la Carrera de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA WEB CON MODELO PREDICTIVO DE DESERCIÓN PARA EL GIMNASIO STRONG 2.0, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, julio 2026

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga, Mg
PRESIDENTE DEL TRIBUNAL

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
FELIX OSCAR FERNANDEZ PEÑA

Ing Félix Oscar Fernández Peña, PhD
PROFESOR CALIFICADOR

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
DENNIS VINICIO CHICAIZA CASTILLO

Ing Dennis Vinicio Chicaiza Castillo, Mg
PROFESOR CALIFICADOR

DEDICATORIA

A mi familia, pilar fundamental en cada etapa de mi vida. Por su apoyo incondicional, su paciencia y por creer en mí incluso en los momentos en que yo mismo dudé. Este logro es tan suyo como mío.

A mis mejores amigos, por estar presentes en los momentos difíciles, por las palabras de aliento cuando el camino se sentía cuesta arriba, y por celebrar cada pequeño avance como si fuera el más grande. Su amistad fue un refugio en este proceso.

Y a mí, por la perseverancia, por levantarme cada vez que quise abandonar, y por elegir seguir adelante. Este trabajo es prueba de que con esfuerzo y dedicación, todo es posible.

AGRADECIMIENTO

Al Ing. Mg. Leonardo Torres, director de este trabajo, por su guía, paciencia y exigencia académica; y al Ing. Mg. Renato Urvina, docente de Desarrollo de Proyectos, por sus enseñanzas y orientación durante el proceso formativo.

Al SOFTWARE SQUAD 404, mi grupo de amigos y compañeros de camino, por las noches de trabajo, las discusiones técnicas y por demostrar que el trabajo en equipo hace más liviano cualquier desafío.

Al Sr. Alex Flores, propietario del gimnasio, por facilitar el acceso necesario para llevar a cabo esta investigación.

A todos quienes contribuyeron de alguna forma a la realización de este trabajo. Cada gesto de apoyo fue significativo.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA i

APROBACIÓN DEL TUTOR II

AUTORÍA III

DERECHOS DE AUTOR IV

APROBACIÓN DEL TRIBUNAL DE GRADO V

DEDICATORIA VI

AGRADECIMIENTO VII

ÍNDICE GENERAL DE CONTENIDOS VIII

ÍNDICE DE TABLAS X

ÍNDICE DE FIGURAS XI

ÍNDICE DE ANEXOS XIII

RESUMEN EJECUTIVO XIV

ABSTRACT XV

I MARCO TEÓRICO 1

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 3

1.3 Fundamentación teórica 16

1.4 Objetivos 20

1.4.1 Objetivo general 20

1.4.2 Objetivos específicos 20

II METODOLOGÍA 20

2.1 Materiales 21

2.2 Métodos 21

2.2.1 Modalidad de investigación y enfoque 21

2.2.2 Población y muestra 22

2.2.3 Recolección de la información 22
2.2.4 Procesamiento y análisis de datos 38

III RESULTADOS Y DISCUSIÓN 41
3.1 Identificar los procesos manuales de gestión de suscripciones y control de asistencia para establecer los requisitos funcionales del nuevo sistema 42
3.2 Determinar el modelo predictivo más adecuado para predecir la deserción de clientes 48
3.3 Implementar el sistema web integrando el modelo predictivo seleccionado para la toma de decisiones 57
3.4 Resultados 86
3.5 Discusión 92

IV CONCLUSIONES Y RECOMENDACIONES 94
4.1 Conclusiones 94
4.2 Recomendaciones 95

REFERENCIAS BIBLIOGRÁFICAS 96

ANEXOS 99

ÍNDICE DE TABLAS

1 Metodologías de Investigación 10
2 Fuentes y Tipos de Datos 11
3 Técnicas de Preprocesamiento 11
4 Ingeniería y Selección de Características 11
5 Modelos de Clasificación 12
6 Modelado Híbrido 12
7 Optimización de Hiperparámetros 13
8 Patrones de Arquitectura 13
9 Lenguajes de Programación Backend 13
10 Frameworks Backend 13
11 Tecnologías Frontend 14
12 Librerías de Machine Learning 14
13 Bases de Datos 15
14 Herramientas de Desarrollo 15
15 Técnicas de Validación 15
16 Métricas de Evaluación 16
17 Interpretabilidad de Modelos (XAI) 16
18 Resultados de la entrevista al administrador del gimnasio Strong 2.0 . 23
19 Análisis de Fiabilidad (Alfa de Cronbach) 25
20 Distribución por rango de edad de los socios 26
21 Tiempo de permanencia de los socios 27
22 Rapidez en el registro de entrada 28
23 Agilidad en la renovación de membresía 29
24 Atención del personal ante consultas 30
25 Claridad en la información de membresías 31
26 Estado de instalaciones y equipos 31
27 Horario de atención 32
28 Seguimiento del progreso físico 33
29 Relación precio - calidad 34
30 Ambiente del gimnasio 35
31 Guía de los entrenadores 36
32 Calificación general de la experiencia 37
33 Matriz FODA de la gestión administrativa en Strong 2.0. 46
34 Lista de requerimientos funcionales. 47
35 Lista de requerimientos no funcionales del sistema. 48

36 Mapeo de variables entre el dataset local (Strong 2.0) y el dataset de Kaggle. 51
37 Resultados del análisis comparativo de modelos. 55
38 Matriz de selección de la biblioteca de optimización de hiperparámetros. 56
39 Hiperparámetros finales del modelo Gradient Boosting obtenidos con Optuna. 57
40 Especificación CU-01: Registrar Asistencia. 59
41 Especificación CU-02: Ejecutar Inferencia de Modelo. 60
42 Especificación CU-03: Analizar Explicabilidad LIME. 61
43 Matriz de selección de la metodología de desarrollo. 64
44 Matriz de selección del framework de desarrollo backend. 64
45 Matriz de selección del framework de desarrollo frontend. 65
46 Planificación de sprints del proyecto. 66
47 Matriz de validación técnica y control de calidad. 78
48 Evaluación comparativa de modelos de Machine Learning. 86
49 Síntesis de impacto operativo: Diagnóstico vs. Implementación. 88
50 Cumplimiento de requisitos funcionales y no funcionales. 92
51 Especificación CU-04: Gestionar Socio. 103
52 Especificación CU-05: Gestionar Membresía. 104
53 Especificación CU-06: Gestionar Alertas de Riesgo. 105

ÍNDICE DE FIGURAS

1 Análisis de categorías emergentes mediante procesamiento de lenguaje natural. 24
2 Nube de palabras generada a partir del procesamiento NLP de la entrevista. 24
3 Distribución porcentual de la Pregunta 1 26
4 Distribución porcentual del tiempo de permanencia de los socios . . . 26
5 Distribución porcentual de la rapidez en el registro de asistencia . . . 27
6 Distribución porcentual de la agilidad en la renovación de membresía . 28
7 Distribución porcentual de la claridad y amabilidad del personal . . . 29
8 Distribución porcentual de la claridad en la información de planes y beneficios 30
9 Distribución porcentual del estado de instalaciones y equipos 31
10 Distribución porcentual de la conveniencia del horario de atención . . 32
11 Distribución porcentual del seguimiento del progreso físico 33
12 Distribución porcentual de la relación beneficio-precio de la membresía 34

13 Distribución porcentual del ambiente motivador del gimnasio 35
14 Distribución porcentual de la disponibilidad y efectividad de los entrenadores 36
15 Distribución porcentual de la experiencia general en el gimnasio . . . 37
16 Análisis de categorías emergentes mediante procesamiento de lenguaje natural. 43
17 BPMN: Proceso de registro de socios (proceso actual) 44
18 BPMN: Proceso de control de asistencia (proceso actual) 44
19 BPMN: Proceso de seguimiento de membresías (proceso actual) . . . 44
20 BPMN: Proceso de cobro de mensualidades (proceso actual) 45
21 Importancia por permutación de las variables del dataset sobre el modelo Gradient Boosting optimizado con Optuna. 53
22 Capacidad de distinción de los modelos (Curvas ROC-AUC). 54
23 Detalle de aciertos y errores por cada modelo (Matriz de Confusión). . 55
24 Diagrama de casos de uso del sistema Strong 2.0. 58
25 BPMN: Proceso de registro de socios (proceso propuesto) 61
26 BPMN: Proceso de control de asistencia (proceso propuesto) 62
27 BPMN: Proceso de seguimiento de membresías (proceso propuesto) . 62
28 BPMN: Proceso de cobro de mensualidades (proceso propuesto) . . . 63
29 Arquitectura de tres capas del sistema Strong 2.0. 68
30 Modelo lógico de datos para la gestión y analítica. 69
31 Flujo de información y procesamiento de predicciones. 70
32 Sprint 5: Módulo de gestión de socios y suscripciones del sistema. Los registros mostrados corresponden al conjunto de datos de demostración, integrado por 150 fichas del gimnasio Strong 2.0 y 4.000 registros del *dataset* público de Kaggle [1], utilizado para ampliar el volumen de entrenamiento del modelo predictivo. 71
33 Sprint 6: Centro de Control del módulo de asistencia del sistema web. 71
34 Sprint 6: Módulo de gestión de pagos: registro de cobros. 72
35 Sprint 6: Módulo de gestión de pagos: historial de transacciones. . . . 72
36 Sprint 7: Actividad de desarrollo: integración del modelo de IA y autenticación JWT. 73
37 Diagrama de secuencia: Registro y análisis predictivo automático. . . 74
38 Interacción del sistema para generar alertas de riesgo. 75
39 Panel de configuración: umbrales de alerta (Riesgo Bajo y Riesgo Crítico), frecuencia de recálculo IA, días de gracia e inactividad manual. 77

40 Panel de configuración: catálogo de membresías disponibles en el gimnasio. 77
41 Resultados de pruebas unitarias en el backend: 35 casos aprobados. . . 79
42 Resumen de cobertura de código tras la ejecución de pruebas unitarias. 80
43 Resultados de pruebas unitarias en el frontend: 16 casos aprobados mediante Vitest. 80
44 Prueba de integración: autenticación JWT, paso 1: login. 81
45 Prueba de integración: verificación 2FA y obtención del token de acceso. 81
46 Prueba de integración: recálculo de predicción de un socio. 82
47 Prueba de integración: consulta de predicción con respuesta JSONB desde PostgreSQL. 82
48 Reporte de cobertura de código mediante Coverage.py: parte 1 de 3 (66 % de cobertura global). 83
49 Reporte de cobertura de código mediante Coverage.py: parte 2 de 3. . 84
50 Reporte de cobertura de código mediante Coverage.py: parte 3 de 3. . 84
51 Prueba de caja negra: dashboard con KPI de riesgo global y alertas de socios en riesgo alto. 85
52 Prueba de caja negra: módulo de análisis predictivo con factores de impacto y candidatos en riesgo. 85
53 Prueba de caja negra: recomendación estratégica generada e historial de ejecuciones del motor predictivo. 86
54 Análisis de discriminación del modelo mediante la Curva ROC-AUC. 87
55 Matriz de Confusión: Validación de aciertos y errores predictivos. . . 87
56 Explicación individual de riesgo de abandono generada por LIME para un socio. 89
57 Predicción del modelo con datos reales de socios del gimnasio Strong 2.0. 89

ÍNDICE DE ANEXOS

A Mockup del dashboard de gestión. 100
B Mockup de la pantalla de gestión de miembros. 100
C Mockup de la pantalla de configuración del sistema. 101
D Mockup de la pantalla de análisis de inteligencia artificial. 101
E Mockup de la pantalla de alertas de riesgo. 102
F Especificación de casos de uso complementarios. 102

RESUMEN EJECUTIVO

El presente trabajo de titulación desarrolló un sistema web con modelo predictivo de deserción para el gimnasio Strong 2.0. El establecimiento gestionaba suscripciones y asistencia con registros físicos y hojas de cálculo desconectadas, lo que generaba una pérdida de productividad superior al 50 %. La investigación buscó responder: ¿de qué manera la implementación de un sistema web con inteligencia artificial optimizará la gestión administrativa y permitirá predecir la deserción de clientes en el gimnasio Strong 2.0?

La investigación adoptó una metodología de investigación aplicada y de campo. El diagnóstico incluyó una entrevista semiestructurada al administrador y una encuesta a socios activos para caracterizar los procesos vigentes. La selección del modelo comparó seis algoritmos mediante validación cruzada estratificada con $k = 5$ y optimización bayesiana de hiperparámetros; la implementación usó Django, Next.js y PostgreSQL.

Se identificaron cuatro procesos manuales sin rastro digital, de los que se establecieron seis requisitos funcionales y cuatro no funcionales. Gradient Boosting fue el modelo de mayor desempeño, con exactitud del 93,48 %, sensibilidad del 84,92 % y ROC AUC de 0,98. Su integración al sistema, complementada con LIME como técnica de inteligencia artificial explicable, generó alertas de deserción con justificación individual por socio. Digitalizó la totalidad de los procesos operativos, eliminó 15 inconsistencias semanales de datos y alcanzó el 66 % de cobertura de código en el backend, cumpliendo todos los requisitos planificados.

Palabras clave: Sistema web, predicción de deserción, aprendizaje automático, Gradient Boosting, LIME, gestión de suscripciones.

ABSTRACT

This study developed a web system with a churn prediction model for Strong 2.0 gym. The establishment managed subscriptions and attendance through physical records and disconnected spreadsheets, generating a productivity loss exceeding 50 %. This study addressed the question: how does the implementation of a web system with artificial intelligence optimize administrative management and enable prediction of customer churn at Strong 2.0 gym?

The research followed an applied and field research methodology. A semi-structured interview and a survey of active members were conducted to diagnose current processes. Six machine learning algorithms were evaluated using stratified k-fold cross-validation ($k = 5$) with Bayesian hyperparameter optimization. The system was built using Django, Next.js, and PostgreSQL.

Four manual processes with no digital trail were identified — member registration, attendance control, membership tracking, and billing — yielding six functional and four non-functional requirements. Gradient Boosting achieved the best performance: accuracy 93.48 %, recall 84.92 %, and ROC AUC 0.98. Integrated with LIME as an explainable AI technique, it generated churn alerts with individual justification per member. It digitalized all operational processes, eliminated 15 weekly data inconsistencies, and reached 82 % code coverage, fulfilling all planned requirements.

Keywords: Web system, churn prediction, machine learning, Gradient Boosting, LIME, subscription management.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

SISTEMA WEB CON MODELO PREDICTIVO DE DESERCIÓN PARA EL GIMNASIO STRONG 2.0

1.1.1 Planteamiento del problema

La industria del fitness ha experimentado un crecimiento notable en los últimos años, impulsado por una creciente conciencia sobre la importancia de la salud física y un estilo de vida saludable, lo que ha llevado a la apertura de numerosos gimnasios y centros deportivos [2, 3, 4]. Este auge responde a una necesidad sanitaria crítica, pues se estima que más de mil millones de adultos sufren problemas relacionados con el sobrepeso, y de ellos, el 30 % presenta obesidad, lo que deriva en enfermedades crónicas [5]. Económicamente, mercados regionales como el de México presentan una tasa de penetración del 3.2 % de la población, atendiendo a millones de usuarios [6]. Este crecimiento se ha visto acentuado por eventos como la pandemia del COVID-19, que obligó a muchos establecimientos a adoptar tecnologías para agilizar sus actividades [2, 7, 6]. En ciudades como Riobamba, Quito, Cajamarca y Naranjal, la amplia oferta de gimnasios ha generado competencia, haciendo que la eficiencia operativa sea esencial [8]. Sin embargo, muchos centros mantienen una dependencia de procesos manuales para la gestión diaria, incluyendo el registro de suscripciones y asistencia [2, 9, 5]. Esta situación genera dificultades para mantener un control adecuado sobre la información, así como retrasos en la actualización de datos, restringiendo la capacidad del personal para adaptarse a la creciente demanda y afectando la experiencia del cliente.

El problema tiene su origen en la persistente utilización de métodos manuales como hojas impresas o Excel desconectados, lo que provoca ineficiencias cuantificables drásticamente en la productividad [2, 8]. Estudios comparativos han demostrado que la implementación de aplicaciones web frente a procesos manuales reduce el tiempo total de gestión de procesos clave (registro, mensualidades y pagos) de 11:01 minutos a 5:13

minutos, representando una mejora global del 52.77 % en el tiempo de respuesta [2]. Desglosando esta optimización, la gestión de pagos logra una reducción del 58.29 %, de 3:31 a 1:28 minutos, mientras que el registro de clientes y suscripciones mejora en un 50.45 % y 49.56 % respectivamente [2]. En términos de rendimiento operativo, un sistema manual permite realizar solo 2 operaciones en un intervalo de 7 minutos, frente a las 5 que permite un sistema automatizado, lo que se traduce en una mejora del 150 % [2]. La gestión mediante sistemas no estructurados provoca que el personal dedique tiempo excesivo a tareas repetitivas, reduciendo la productividad y generando frustración.

Consecuentemente, estas limitaciones impactan directamente la competitividad y la satisfacción de los usuarios [3, 7, 6]. En cadenas en crecimiento, la falta de procesos estructurados ha provocado aumentos bimestrales en las quejas de usuarios que oscilan entre el 28 % y el 44 % en diversas regiones [6]. La gestión ineficiente genera una "caja negra" en la retención: se ha detectado que hasta un 41 % de las cancelaciones de membresías no tienen un motivo especificado debido a la falta de captura de datos de salida [6]. La imposibilidad de generar reportes detallados y la insatisfacción frente a un servicio lento disminuyen la fidelidad, afectando la reputación del gimnasio. La situación evidencia la necesidad de modernizar los procesos para garantizar una operación eficiente y un acceso confiable a la información, factores clave para la sostenibilidad.

Esta problemática se manifiesta de forma crítica en el gimnasio **Strong 2.0**, un centro de acondicionamiento físico con 5 años de trayectoria en la ciudad de Ambato, ubicado estratégicamente en el sector Mall de los Andes. Actualmente, la empresa opera con una matriz que gestiona una población de aproximadamente 200 socios, de los cuales se estima que solo el 60 % mantiene una suscripción activa de manera regular.

A pesar de su crecimiento sostenido desde su fundación en 2018, el gimnasio carece de una herramienta centralizada, dependiendo de registros físicos y hojas de cálculo aisladas para controlar el flujo diario. Se ha observado que, en promedio, un 30 % de los nuevos inscritos deserta antes de cumplir el tercer mes de membresía, una cifra alineada con la tendencia regional de abandono prematuro [10]. La imposibilidad de

correlacionar la frecuencia de asistencia diaria con la vigencia de las suscripciones crea un vacío de información; el personal administrativo no cuenta con alertas que identifiquen cuándo un socio ha dejado de asistir por más de 7 días consecutivos, lo que imposibilita la detección temprana de usuarios en riesgo de abandono. Esta falta de seguimiento proactivo resulta en una tasa de deserción que no puede ser gestionada ni prevenida, afectando directamente la estabilidad financiera y el tiempo de vida del cliente dentro de la empresa.

El problema central es que el aumento de socios expuso las limitaciones de la gestión manual. Esto provocó una pérdida de productividad superior al 50 % y limitó la capacidad de asegurar la satisfacción del cliente. Para atender esta situación, se propone implementar un sistema web que automatice la administración e integre modelos predictivos de deserción.

Frente a esta problemática, se plantea la siguiente pregunta de investigación: ¿De qué manera la implementación de un sistema web integral con inteligencia artificial optimizará la gestión administrativa y permitirá predecir la deserción de clientes en el gimnasio Strong 2.0?

1.2 Antecedentes investigativos

En [11] se presenta un modelo comparativo para predecir la deserción de clientes en el sector de telecomunicaciones, una industria que sufre tasas anuales de churn de hasta el 30 %. La metodología SEMMA (Muestreo, Exploración, Modificación, Modelado y Evaluación) se aplicó sobre un dataset de Kaggle con más de 7.000 clientes y 21 atributos que incluyen datos demográficos, servicios recibidos y cargos totales. Se compararon ocho algoritmos de clasificación, incluyendo Random Forest, Árboles de Decisión, AdaBoost, SGD, Regresión Logística, KNN, CatBoost y Gradient Boosting. La contribución principal del estudio es la identificación de los factores clave que influyen en la deserción y la obtención de un modelo eficiente de predicción, lo que proporciona un referente metodológico útil para proyectos de predicción de churn.

En [12] se realiza una revisión sistemática de 212 artículos publicidos entre 2015 y 2023 sobre predicción de churn usando machine learning, analizando exhaustivamente las etapas de preprocesamiento de datos, ingeniería de características y validación de modelos. Los métodos de Ensemble y Deep Learning generalmente superan en rendimiento predictivo a los modelos más simples; asimismo, se destaca la importancia de incluir características demográficas, de uso, de comportamiento e interacción social. Como principal contribución, se recomienda priorizar el uso de técnicas de ensamblaje y aprendizaje profundo junto con métodos explicables (XAI), además de abogar por el uso de métricas de evaluación basadas en ganancias para abordar la brecha investigativa sobre la rentabilidad de los modelos.

En [13] se desarrolla un modelo predictivo de churn para miembros de gimnasios usando redes neuronales artificiales tipo perceptrón multicapa (MLP) con retropropagación, con especial énfasis en la selección de características mediante revisión bibliográfica y métodos de filtrado. La investigación incorpora el concepto psicológico de formación de hábitos como variable de entrada, encontrando una correlación negativa entre la asistencia y la tendencia a la deserción. El modelo alcanza métricas de desempeño excepcionales: 92.1 % de exactitud, 89.1 % de sensibilidad y 93.8 % de especificidad, demostrando la efectividad de integrar principios psicológicos en modelos de machine learning para la retención de clientes en el sector fitness.

En [10] se estudia la predicción de deserción en un gimnasio portugués con una muestra de 5.209 miembros, evaluando el rendimiento de algoritmos de machine learning utilizando datos históricos de uso de las instalaciones. El análisis identifica que el Gradient Boosting Classifier obtuvo el mejor rendimiento general, con exactitud de 0.955 y sensibilidad de 0.986, mientras que Random Forest Classifier fue superior prediciendo la no-deserción, con especificidad de 0.790. Las variables más relevantes resultaron ser "días sin asistencia", "tiempo total de permanenciaƶ "monto total facturado". Se destaca la utilidad de los árboles de decisión para identificar perfiles de riesgo y fundamentar medidas preventivas personalizadas.

En [14] se presenta SensorChat, el primer sistema de preguntas y respuestas integral diseñado para monitoreo diario utilizando series temporales de datos de sensores de alta frecuencia y larga duración. La propuesta implementa un pipeline innovador de tres etapas: descomposición de preguntas, consulta de datos y ensamblaje de respuestas, aprovechando modelos de lenguaje para interpretar consultas y generar respuestas precisas. El sistema alcanza hasta un 93 % más de precisión en preguntas cuantitativas respecto a sistemas existentes y demuestra efectividad en preguntas cualitativas, representando un avance significativo en la interacción natural con datos sensoriales complejos.

En [15] se analiza el impacto de la influencia social en la probabilidad de churn mediante métricas de red social denominadas ïnfluenciaz çonformidad", desarrolladas a partir de datos de redes de llamadas telefónicas (CDR). La investigación aplica diversos algoritmos de clasificación supervisada y evalúa su desempeño con métricas como AUC, precisión y F1-score. Los resultados demuestran que la incorporación de estas variables de red social mejora significativamente el rendimiento de todos los modelos, siendo Gradient Boosting el de mejor desempeño, validando así el valor del análisis de redes sociales en la predicción de deserción clientelar.

En [16] se presenta un estudio experimental de tres meses con 19 participantes que evalúa el impacto de Test-Driven Development (TDD) frente a Test-Last Development (TLD) en un proyecto industrial real para una empresa de software del sector energético. La investigación organizó a los participantes en tres bloques de competencia, denominados Mixed, Intermediate y Novice, y demuestra que TDD reduce significativamente la cantidad de errores en 1.8 bugs y mejora la calidad de los tests en 5 puntos porcentuales de cobertura frente a TLD. También se evidencia que TDD tiene una curva de aprendizaje más pronunciada pero reduce la brecha de competencia entre desarrolladores.

En [17] se analiza la aplicación de TDD en una aplicación web cross-platform

que incluye backend con webservice y frontend como Progressive Web Application. La investigación demuestra que TDD mejora la calidad del código del backend y permite al frontend alcanzar mejores puntuaciones en la herramienta Lighthouse de evaluación de desempeño. Se destaca la importancia de planificar funcionalidades antes de codificar para prevenir errores y optimizar el proceso de desarrollo, estableciendo pautas prácticas para la implementación exitosa de TDD en proyectos web full-stack.

En [18] se propone un enfoque de transporte óptimo regularizado para análisis de datos funcionales, aproximando mapas estocásticos entre espacios de funciones usando operadores de Hilbert-Schmidt. El método garantiza existencia, unicidad y consistencia del estimador para el espacio de Hilbert-Schmidt de operadores lineales compactos. La validación con datasets sintéticos y trayectorias de robots demuestra su eficacia en adaptación de dominio y análisis de datos de alta dimensión, contribuyendo al avance de técnicas matemáticas sofisticadas para el procesamiento de datos funcionales complejos en aplicaciones de machine learning.

En [19] se desarrolla un modelo de predicción de churn usando clasificadores de machine learning que considera características demográficas y de comportamiento. La investigación aborda el problema del desequilibrio de clases mediante la técnica SMOTEENN, que combina sobremuestreo con edición de vecinos más cercanos. Los resultados muestran que en el dataset desequilibrado, CatBoost tuvo el mejor desempeño con 80 % de exactitud, mientras que en el dataset equilibrado con SMOTEENN, K-Nearest Neighbors emergió como el modelo superior con 98.1 % de exactitud, validando el papel crucial del preprocesamiento de datos en el rendimiento predictivo.

En [20] se analiza la predicción de churn en modelos de negocio basados en suscripción, comparando nueve algoritmos con 21 características que cubren datos demográficos, de facturación, patrones de uso y relaciones con el cliente. A través de un procedimiento riguroso que incluyó preparación de datos, ingeniería y selección de características, se identificó que Logistic Regression, Gradient Boosting con SMOTE

y Redes Neuronales sobresalieron, siendo Logistic Regression el de mayor precisión con 79.6 %. Se evidencia que SMOTE no mejoró el rendimiento en este contexto y se resalta la importancia de seleccionar algoritmos según el dominio específico del negocio.

En [21] se implementa un sistema de gestión inteligente de churn en bancos utilizando machine learning, con el objetivo de prevenir la atrición clientelar mediante el monitoreo y análisis automatizado de las razones de deserción. La metodología comprendió cuatro fases: preparación y procesamiento de datos, extracción de características, dataset y pruebas, y evaluación de rendimiento. Se compararon seis algoritmos de clasificación. Random Forest fue seleccionado como el modelo de mejor rendimiento con 87 % de precisión y 85 % de precisión en predicción positiva, demostrando la efectividad del enfoque para mejorar la satisfacción clientelar mediante la retención proactiva.

En [22] se propone un modelo interpretable para predecir la aglomeración en gimnasios, abordando la problemática de hacinamiento que genera frustración e insatisfacción en usuarios. La investigación utiliza LIME para explicabilidad y compara el rendimiento de modelos individuales como LSTM, RNN, TCN, CNN y RandomForestRegressor, así como de ensembles. RandomForestRegressor demostró ser el mejor modelo individual con MAE de 4.34, RMSE de 6.41 y R^2 de 0.92, mientras que el ensemble LSTM+TCN superó a otros ensembles, validando la utilidad de combinaciones de modelos para la predicción de multitudes en espacios deportivos.

En [2] se desarrolla una aplicación web para la gestión de clientes y control de pagos de mensualidades del gimnasio “Team Legión” en Riobamba, utilizando la metodología Proceso Unificado Ágil (AUP) con fases de Concepción, Elaboración, Construcción y Transición. El sistema permite administrar membresías, pagos, control de asistencia, profesores, torneos y reportes mediante una plataforma centralizada. La evaluación bajo el estándar ISO/IEC 25010 demostró mejoras significativas en el comportamiento temporal y eficiencia operativa, estableciendo un referente para la digitalización de procesos en el sector fitness ecuatoriano.

En [3] se crea un sistema web y aplicación móvil para la gestión operativa de un centro de crossfit en Quito, utilizando la metodología SCRUM con división del trabajo en Sprints. El desarrollo se centró en el backend con tecnologías como Next.js, Prisma, PostgreSQL y el patrón MVC, incluyendo pruebas unitarias con unittest, de rendimiento con Locust y de estrés. El sistema integra gestión de clientes, planificación de clases y control de pagos, optimizando procesos administrativos y facilitando la gestión diaria del centro mediante una plataforma robusta y escalable.

En [9] se diseña una aplicación web para administrar actividades administrativas y operativas del gimnasio Hammer Gym en Naranjal, utilizando la metodología Proceso Unificado Racional (RUP) que permite validaciones coherentes en cada iteración. Desarrollada con Python, Django y PostgreSQL, la plataforma incluye módulos para gestión de clientes, empleados, insumos, inventario, proveedores, cobro de mensualidades y ventas de insumos. La implementación solucionó las inconsistencias del manejo manual de información, mejorando sustancialmente la organización interna y los procesos de gestión.

En [8] se desarrolla un sistema web en .NET y aplicación móvil nativa Android para la gestión del Gimnasio Anahí en Quito, mediante un proceso comparativo que determinó la superioridad de metodologías ágiles para el proyecto. La solución combina SCRUM para gestión del proyecto y Extreme Programming (XP) para ejecución del desarrollo, centralizando información de clientes, clases y pagos en una plataforma integrada que incluye reserva de cupos en línea. El sistema facilita la administración y seguimiento de miembros, estableciendo las bases para la gestión digitalizada de establecimientos fitness.

En [23] se estudia el impacto de un aplicativo software en la gestión de clientes de un gimnasio en Cajamarca, desarrollado con metodología ágil SCRUM y tecnologías ASP.Net en programación N capas. La investigación demostró que tras la implementa-

ción se alcanzó un 87.6 % de aceptación, reflejando mejora sustancial en seguridad de datos, tiempo de registro y satisfacción del cliente compared to los procesos manuales anteriores que resultaban en pérdida de clientes potenciales y baja satisfacción en la atención.

En [7] se desarrolla un sistema web gerencial para la optimización financiera de gimnasios, adoptando metodología SCRUM con artefactos como Recopilación de Requerimientos e Historias de Usuario. La arquitectura MVC asegura separación de preocupaciones entre interfaz, datos y lógica de control, utilizando React para la interfaz de usuario con librerías como Formik y Firebase. El sistema integra gestión de datos financieros, reportes y análisis para apoyar la toma de decisiones estratégicas, contribuyendo a mejorar la rentabilidad mediante administración financiera más eficiente y basada en datos.

En [4] se crea una plataforma de gestión para gimnasio en Pindal, Loja, utilizando una adaptación de metodología ágil basada en Extreme Programming (XP) y el framework Django con Python implementando patrón MVC. La aplicación busca optimizar la gestión y consulta de información con módulos clave como seguimiento de miembros con atributos de suscripción y vencimiento, y sistema de notificaciones para cuotas vencidas y eventos diarios. La implementación durante 2022-2023 demostró funcionamiento satisfactorio y mejoras en el control operativo del establecimiento.

En [6] se implementa un sistema para gestionar procesos en Station 24 Fitness, abordando la problemática de falta de organización y estructura derivada del crecimiento exponencial del gimnasio. La metodología de intervención fusiona elementos de Kubr (preparativos y planeación), Six Sigma con DMAIC (mejorar y controlar) y Design Thinking (evaluación), creando un “traje a la medida” para la cultura empresarial. El primer proceso implementado se basó en la medición de satisfacción del cliente, permitiendo mejorar la experiencia de usuario y optimizar la gestión operativa mediante un enfoque sistemático.

La literatura demuestra grandes avances en el uso de *Machine Learning* para predecir deserciones y en buenas prácticas de software. Sin embargo, todavía falta integrar ambos enfoques en sistemas web creados específicamente para administrar gimnasios. Analizar estos estudios permite consolidar una base técnica sólida para desarrollar Strong 2.0: una solución que organice las operaciones y anticipe el comportamiento de los clientes.

El análisis comparativo de los estudios revisados permitió identificar patrones metodológicos, tecnológicos y analíticos recurrentes en investigaciones orientadas al desarrollo de sistemas predictivos basados en machine learning. Para garantizar una visión estructurada, los artículos fueron clasificados según su enfoque metodológico, tipo de datos, técnicas de preprocesamiento, modelos empleados, arquitecturas de software, tecnologías implementadas y métodos de validación. Los trabajos revisados convergen en tres prácticas dominantes: la validación empírica sobre datos de dominio específico como metodología preferida, el uso de métodos de ensamblaje y deep learning para superar modelos individuales en predicción de abandono, y la adopción de Python con arquitecturas MVC como base tecnológica para integrar componentes web con pipelines de machine learning.

Tabla 1: Metodologías de Investigación

Metodología de Investigación	**Artículos**
Revisión Sistemática	[12]
Estudio de Caso / Exploratoria	[11, 13, 19, 20, 21, 15, 10, 16, 24, 22, 17, 25, 18, 14]

Como se muestra en la Tabla 1, los estudios de caso y enfoques exploratorios constituyen la metodología predominante en catorce de las investigaciones revisadas, mientras que solo un estudio emplea revisión sistemática. La razón es estructural: la predicción de deserción depende de datos propietarios y condiciones operativas que no pueden capturarse mediante revisiones de literatura. Sin validación empírica en el dominio, los modelos carecen de relevancia práctica.

Tabla 2: Fuentes y Tipos de Datos

Fuente/Tipo de Datos	**Artículos**
Kaggle	[11, 19, 20]
Datos Propietarios	[13, 10]
Registros de Llamadas (CDR)	[15]
Sensores Móviles	[24, 14]
Datos Industriales	[16]
Entrevistas y Encuestas	[25]

En la Tabla 2 se identifica a Kaggle como la fuente de datos más recurrente en tres estudios, lo que refleja su accesibilidad para el benchmarking inicial de modelos predictivos. La diversidad de fuentes restantes, que abarca desde datos propietarios hasta sensores móviles, confirma que no existe una estrategia universal de recopilación: la naturaleza del problema determina la fuente.

Tabla 3: Técnicas de Preprocesamiento

Técnica de Preprocesamiento	**Artículos**
SMOTE	[12, 20, 21]
SMOTEENN	[19]
Método de Filtro (Spearman, Cramer's V)	[13]

La Tabla 3 revela que SMOTE emerge como la técnica de preprocesamiento más utilizada en tres investigaciones. Su adopción superior frente a alternativas como SMOTEENN responde a un balance práctico entre coste computacional y eficacia: genera muestras sintéticas de la clase minoritaria sin las pérdidas de información que introduce la edición de vecinos. En dominios con desbalance natural entre clientes activos e inactivos, esta ventaja es determinante.

Tabla 4: Ingeniería y Selección de Características

Técnica	**Artículos**
Ingeniería de Características	[15, 20, 12]
Selección de Características	[13, 20, 12]
Análisis RFM	[12]

Según la Tabla 4, la ingeniería y selección de características muestran igual prevalencia en tres estudios cada una. Esta simetría no es casual: crear características relevantes a partir de datos transaccionales y luego filtrar las que aportan señal son operaciones distintas pero inseparables. La presencia del análisis RFM en uno de los estudios ilustra este principio con claridad, al transformar registros de compra en indicadores de comportamiento directamente interpretables por los modelos.

Tabla 5: Modelos de Clasificación

Tipo de Modelo	**Artículos**
Interpretables (Regresión Logística, Árbol Decisión)	[11, 20, 21, 12, 10]
No interpretables (Random Forest, CatBoost, ANN)	[11, 19, 20, 21, 13, 12, 15]

Como se presenta en la Tabla 5, los modelos no interpretables superan en uso a los interpretables por siete estudios frente a cinco. La diferencia es estrecha, y eso es significativo: refleja que el sector no puede permitirse optimizar solo el rendimiento predictivo. Cuando el resultado de un modelo determina una acción de retención sobre un cliente real, la transparencia deja de ser opcional.

Tabla 6: Modelado Híbrido

Técnica Híbrida	**Artículos**
Ensemble	[11, 12, 22]
Deep Learning	[13, 12, 24, 22, 18, 14]
Logit Leaf	[12]

La Tabla 6 evidencia que el Deep Learning lidera las técnicas híbridas con seis estudios, seguido por Ensemble en tres investigaciones. La supremacía del aprendizaje profundo refleja su capacidad para capturar dependencias temporales y patrones multimodales que los modelos tabulares tradicionales no modelan. La aparición de técnicas como Logit Leaf —presente en un único estudio— indica que el campo sigue explorando combinaciones que traduzcan capacidad predictiva en reglas interpretables.

Tabla 7: Optimización de Hiperparámetros

Técnica de Optimización	**Artículos**
GridSearchCV	[19, 20]
Manual	[11, 13]

En la Tabla 7 se observa que GridSearchCV y la optimización manual presentan igual adopción con dos estudios cada una. La paridad revela una tensión real: GridSearchCV garantiza exhaustividad pero escala mal con el número de parámetros, mientras que la optimización manual permite incorporar conocimiento del dominio a costa de reproducibilidad. La elección entre ambos depende directamente de los recursos computacionales y la complejidad del modelo evaluado.

Tabla 8: Patrones de Arquitectura

Patrón de Arquitectura	**Artículos**
MVC	[2, 3, 7, 4, 17]
n-capas	[2]
API RESTful	[3, 7]

Como se detalla en la Tabla 8, el patrón MVC domina claramente (5 estudios) sobre otras arquitecturas, demostrando su madurez y amplia adopción en desarrollo web empresarial.

Tabla 9: Lenguajes de Programación Backend

Lenguaje	**Artículos**
Python	[11, 20, 4, 17]
JavaScript/Node.js	[3, 7]
Java	[2]

Tabla 10: Frameworks Backend

Framework	**Artículos**
Django	[4, 17]
Next.js	[3, 7]
Spring Boot	[2]

Como se detalla en las Tablas 9 y 10, Python y Django constituyen el stack backend predominante en cuatro y dos estudios respectivamente. Su ventaja frente a JavaScript/Node.js o Java reside en la continuidad del entorno: el mismo lenguaje que entrena los modelos ejecuta la API que los sirve, eliminando la capa de traducción entre componentes. La distribución equilibrada de frameworks restantes confirma que fuera de Python, cada solución domina exclusivamente dentro de su propio ecosistema.

Tabla 11: Tecnologías Frontend

Tecnología	**Artículos**
React	[3, 7, 17]
HTML5/CSS/JavaScript	[11, 3, 7, 4, 2, 17]
Tailwind CSS	[2, 3, 7]
Bootstrap	[4]

Según la Tabla 11, React emerge como el framework frontend predominante en tres estudios. Su modelo de componentes facilita la construcción de dashboards interactivos con actualización selectiva del DOM, una necesidad concreta en interfaces que visualizan datos predictivos en tiempo real. El desplazamiento de Bootstrap por Tailwind CSS en los proyectos más recientes refleja la misma lógica: mayor control granular sobre el diseño sin depender de estilos preestablecidos.

Tabla 12: Librerías de Machine Learning

Librería	**Artículos**
Scikit-learn	[11, 19, 20, 21, 12]
Deep Learning (TensorFlow/PyTorch)	[24, 14, 18]
XGBoost/CatBoost	[11, 19, 20]

Como se presenta en la Tabla 12, Scikit-learn consolida su posición como librería estándar en cinco estudios. Su dominio se explica por la API uniforme que expone: preprocesamiento, entrenamiento, validación cruzada y métricas comparten la misma interfaz, lo que reduce el coste de integración. El crecimiento paralelo de TensorFlow/PyTorch y las librerías de boosting no desafía esa posición, sino que la complementa: Scikit-learn estructura el pipeline; las otras proveen los estimadores de mayor capacidad.

Tabla 13: Bases de Datos

Base de Datos	**Artículos**
PostgreSQL	[3, 7, 17]
MySQL	[2]
SQLite	[4]

En la Tabla 13 se observa que PostgreSQL es claramente preferida en tres estudios. Su ventaja sobre MySQL reside en el soporte nativo de tipos JSON, rangos temporales y extensiones estadísticas que resultan útiles para almacenar resultados de inferencia junto a los datos transaccionales. SQLite, presente en un único proyecto de menor escala, confirma que la elección del motor refleja directamente los requisitos de concurrencia y volumen de cada implementación.

Tabla 14: Herramientas de Desarrollo

Herramienta	**Artículos**
Figma	[3, 7]
GitHub	[3, 7]
Vercel	[3, 7]

La Tabla 14 indica que las herramientas modernas de desarrollo muestran adopción concentrada en proyectos específicos, con Figma, GitHub y Vercel apareciendo consistentemente en los mismos estudios (2 estudios cada una). Esto sugiere la formación de stacks de desarrollo integrados donde diseño, control de versiones y despliegue están estrechamente alineados. La relativa baja mención general indica que muchas publicaciones no documentan exhaustivamente su toolchain de desarrollo.

Tabla 15: Técnicas de Validación

Técnica de Validación	**Artículos**
Hold-out	[10]
Validación cruzada	[19, 20, 13, 12]

La Tabla 15 muestra que la validación cruzada es claramente predominante en cuatro estudios frente al hold-out en uno. En predicción de deserción, donde los conjuntos de

datos raramente superan los pocos miles de registros, el hold-out sacrifica demasiada información de entrenamiento para generar una estimación confiable. La validación cruzada resuelve ese dilema al rotar el subconjunto de evaluación, aprovechando cada observación tanto para ajustar como para validar el modelo.

Tabla 16: Métricas de Evaluación

Métrica	**Artículos**
Matriz de confusión	[21, 12]
ROC-AUC	[11, 19, 12, 15]
Métricas de negocio	[12]

En la Tabla 16 se identifica a ROC-AUC como la métrica más utilizada en cuatro estudios, superando claramente a la matriz de confusión y las métricas de negocio. Su adopción refleja una necesidad concreta: en clasificación desbalanceada, la exactitud global es engañosa, y ROC-AUC captura el rendimiento del modelo a lo largo de todos los umbrales de decisión posibles. La baja adopción de métricas de negocio, presente en un único estudio, es la observación más relevante del conjunto: un modelo técnicamente sólido que no se evalúa en términos de retención o ingresos recuperados ofrece garantías incompletas sobre su valor real para la organización.

Tabla 17: Interpretabilidad de Modelos (XAI)

Técnica XAI	**Artículos**
LIME	[12, 22]

La Tabla 17 indica que LIME se posiciona como estándar emergente de explicabilidad en el dominio, respaldado por dos estudios del área.

1.3 Fundamentación teórica

Fundamentos de Machine Learning para la Predicción de Churn en Gimnasios

El machine learning constituye el pilar fundamental para el desarrollo de sistemas predictivos de abandono de clientes en el sector fitness. El contraste con los enfoques

estadísticos tradicionales es directo. El aprendizaje automático permite capturar patrones no lineales y relaciones complejas en los datos de comportamiento de usuarios que los modelos clásicos no pueden representar [11, 12].

La literatura especializada demuestra que el problema de predicción de churn en servicios de fitness presenta características particulares. [13] y [10] destacan que la naturaleza temporal y de comportamiento de los datos de asistencia exige modelos capaces de adaptarse a patrones estacionales y cambios en los hábitos de ejercicio. Esta complejidad justifica la implementación de arquitecturas de machine learning que trasciendan los modelos básicos y capturen la complejidad del comportamiento del usuario.

Arquitectura Integrada de Procesamiento de Datos para Entornos Fitness

El tratamiento de datos en entornos de gimnasio presenta desafíos específicos que requieren una pipeline sólida de preprocesamiento. [12] y [20] evidencian que la calidad de los datos de asistencia y membresía es determinante para el rendimiento predictivo en este sector. La ingeniería de características debe considerar variables particulares del dominio fitness, como frecuencia de asistencia, consistencia en horarios, participación en clases grupales y relación costo-beneficio percibida por el usuario.

La selección de atributos relevantes adquiere dimensiones estratégicas en este contexto. [15] sostiene que en entornos de gimnasios, factores como la regularidad en la asistencia y la participación en servicios adicionales muestran mayor poder predictivo que variables demográficas tradicionales. Esta perspectiva orienta el diseño de características hacia indicadores de comportamiento específicos del dominio de fitness.

Enfoques de Modelado para la Predicción de Abandono en Servicios por Sus-

cripción

Los algoritmos de clasificación representan la base del modelado predictivo de churn, pero requieren adaptaciones específicas para el contexto de gimnasios. [19] y [21] advierten sobre la necesidad de adaptar estas técnicas al desbalance característico de los datos de abandono en servicios recurrentes, donde los casos positivos típicamente constituyen una minoría. Esta particularidad exige estrategias especializadas de muestreo y ponderación para mejorar el rendimiento de los clasificadores.

El modelado híbrido emerge como respuesta a las limitaciones de los enfoques únicos en entornos fitness. [11] y [24] demuestran que la combinación de múltiples algoritmos permite capturar diferentes facetas del comportamiento de abandono.

Arquitectura de Software para Sistemas de Gestión Fitness con Capacidades Predictivas

El diseño de sistemas para gestión de gimnasios que integren capacidades predictivas demanda arquitecturas escalables y mantenibles. [2] y [3] subrayan la importancia de separar claramente los componentes de procesamiento de datos, modelado predictivo e interfaces de usuario. Esta separación facilita la evolución independiente de cada módulo según las necesidades cambiantes del negocio.

La selección de tecnologías debe priorizar la compatibilidad entre componentes de gestión administrativa y análisis predictivo. [4] documenta que frameworks que permitan APIs flexibles y almacenamiento eficiente de datos temporales son esenciales para entornos de gimnasios con alta variabilidad de uso. Esta consideración es determinante para arquitecturas de backend que integran componentes de aplicación web con pipelines de aprendizaje automático de manera eficiente.

Validación y Evaluación en Contextos Operativos Reales

La medición del rendimiento de modelos predictivos en entornos fitness requiere métricas alineadas con los objetivos de negocio específicos. [10] y [21] enfatizan que en predicción de churn para gimnasios, métricas como sensibilidad y valor predictivo positivo tienen mayor relevancia operativa que la precisión global, ya que es más costoso no identificar a un cliente que probablemente abandone que clasificar erróneamente a

uno que se mantendrá.

La validación de modelos debe reflejar las condiciones reales de operación de un gimnasio. [20] propone esquemas de evaluación temporal que simulan el despliegue progresivo, un enfoque especialmente relevante en contextos donde los patrones de uso exhiben variaciones estacionales.

Inteligencia Artificial Explicable (XAI) para la Interpretación de Predicciones

Los modelos de machine learning complejos como el Gradient Boosting presentan una limitación en entornos operativos: su naturaleza de "caja negra" dificulta comprender las razones detrás de cada predicción. La Inteligencia Artificial Explicable (XAI) surge como respuesta, proporcionando métodos que permiten interpretar las decisiones de modelos complejos en términos comprensibles para usuarios no técnicos [12, 22].

LIME (*Local Interpretable Model-agnostic Explanations*) es una técnica XAI que genera, para cada predicción individual, una explicación local que identifica qué variables impulsaron ese resultado específico. A diferencia de métricas globales de importancia de características, LIME opera a nivel de instancia: en el contexto de Strong 2.0, el administrador no recibe solo un porcentaje de riesgo, sino una guía de intervención concreta que identifica qué factores, como la frecuencia de asistencia, la antigüedad o el tipo de membresía, motivaron esa predicción específica.

Analizar los antecedentes y el marco teórico confirma que este proyecto es técnica y metodológicamente viable. Los estudios revisados demuestran que el *Machine Learning* es una herramienta eficaz para predecir la deserción de clientes, destacando especialmente el potencial de algoritmos como *Gradient Boosting* y *Random Forest*. Todo este respaldo científico justifica el desarrollo del sistema Strong 2.0, integrando inteligencia artificial para anticipar y prevenir el abandono de los socios.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema web para la gestión de suscripciones y asistencia en el gimnasio Strong 2.0 Gym, durante el período académico Enero 2026 - julio 2026.

1.4.2 Objetivos específicos

1. Identificar los procesos manuales de gestión de suscripciones y control de asistencia para establecer los requisitos funcionales del nuevo sistema.
2. Determinar el modelo predictivo más adecuado, mediante el análisis comparativo de algoritmos, para anticipar la deserción de clientes en el gimnasio “Strong 2.0”.
3. Implementar el sistema web que centralice la gestión administrativa e integre el modelo predictivo seleccionado para la toma de decisiones.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

Para el desarrollo de la presente investigación y el diagnóstico del estado actual del gimnasio "Strong 2.0", se emplearon los siguientes instrumentos y herramientas:

- **Encuestas:** Diseñadas en la plataforma Google Forms, dirigidas a los socios activos para detectar puntos críticos de deserción.
- **Entrevistas:** Guía de entrevista estructurada aplicada al administrador del gimnasio para conocer los procesos de gestión de suscripciones y control de acceso.

2.2 Métodos

2.2.1 Modalidad de investigación y enfoque

La presente investigación se define como aplicada, ya que su propósito fundamental es la resolución de un problema práctico identificado en la gestión del gimnasio “Strong 2.0” mediante la creación de un sistema web con analítica predictiva.

Se emplea una modalidad de campo, debido a que la recolección de la información primaria se realizó directamente en el entorno real del establecimiento mediante encuestas y entrevistas, permitiendo un diagnóstico preciso de los procesos manuales.

La investigación utiliza un enfoque mixto. La parte cuantitativa permite procesar estadísticamente las encuestas a los socios. Por otro lado, la fase cualitativa ayuda a interpretar a fondo las fallas operativas detalladas por la administración en las entrevistas. Combinar ambas perspectivas es fundamental para estructurar la lógica del sistema y entrenar correctamente el modelo de Inteligencia Artificial.

2.2.2 Población y muestra

Población: La población es finita y está constituida por 200 clientes, tanto activos como registrados históricamente, del gimnasio Strong 2.0.

Muestra: Se aplica un muestreo probabilístico aleatorio simple. Considerando un nivel de confianza del 95 % ($Z = 1{,}96$), un error del 5 % ($e = 0{,}05$) y una probabilidad de éxito y fracaso del 50 % ($p = 0{,}5, q = 0{,}5$), se obtiene una muestra de $n \approx 132$ socios.

$$n = \frac{N \cdot Z^2 \cdot p \cdot q}{e^2 \cdot (N-1) + Z^2 \cdot p \cdot q} \tag{1}$$

$$n = \frac{200 \cdot (1{,}96)^2 \cdot 0{,}5 \cdot 0{,}5}{(0{,}05)^2 \cdot (200-1) + (1{,}96)^2 \cdot 0{,}5 \cdot 0{,}5} \approx 132$$

2.2.3 Recolección de la información

- **Entrevista:** Se aplicó al administrador principal del gimnasio, con el objetivo de identificar las falencias en el registro manual de suscripciones.

Tabla 18: Resultados de la entrevista al administrador del gimnasio Strong 2.0

Nro.	Pregunta	Respuesta	Conclusión
1	¿Podría describir el flujo de los procesos de suscripción y asistencia?	Hasta hace dos semanas se usaban hojas físicas y cuadernos. Recién se migra a Excel, pero la transición es desordenada.	Transición ineficiente de procesos 100 % manuales a digitalización parcial.
2	¿Qué problemáticas identifica en el sistema actual?	Duplicidad de información e inconsistencia en pagos (15 casos semanales). Los registros físicos se dañan o pierden.	Alta vulnerabilidad y falta de integridad en los datos operativos.
3	¿Qué datos históricos mantienen y en qué formato?	18 meses de datos en papel, pero solo 15 días en Excel. Consultar la información física es difícil y lento.	Disponibilidad limitada de datos digitalizados para el modelo predictivo.
4	¿Qué variables cree que influyen en la deserción de clientes?	Inasistencia en días pico (lunes y miércoles), quejas sobre equipos y solicitud frecuente de descuentos.	Identificación de patrones conductuales y económicos como indicadores de Churn.
5	¿Qué funcionalidades considera prioritarias en el sistema?	Un sistema integrado para 200 clientes, control automático de vencimientos y alertas por inasistencia.	Necesidad de centralización y automatización de alertas para retención.

Información técnica complementaria:

- **Volumen actual de clientes:** 200
- **Formatos de almacenamiento:** Hojas físicas y Excel (transición reciente)
- **Procesos más demandantes:** Control manual de asistencia y vencimiento de suscripciones (2–3 horas/día)
- **Disponibilidad de datos históricos:** 18 meses en papel, 2 semanas en Excel

Procesamiento y Análisis de la Entrevista:

Para el procesamiento de la información obtenida en la entrevista con el administrador del Gimnasio Strong 2.0, se desarrolló un script en el lenguaje de programación Python, aplicando técnicas de minería de texto y procesamiento de lenguaje natural (NLP). El proceso consistió en la tokenización del discurso,

filtrado de palabras vacías (*stopwords*) y una categorización semántica por niveles de enraizamiento, medida por frecuencia de mención.

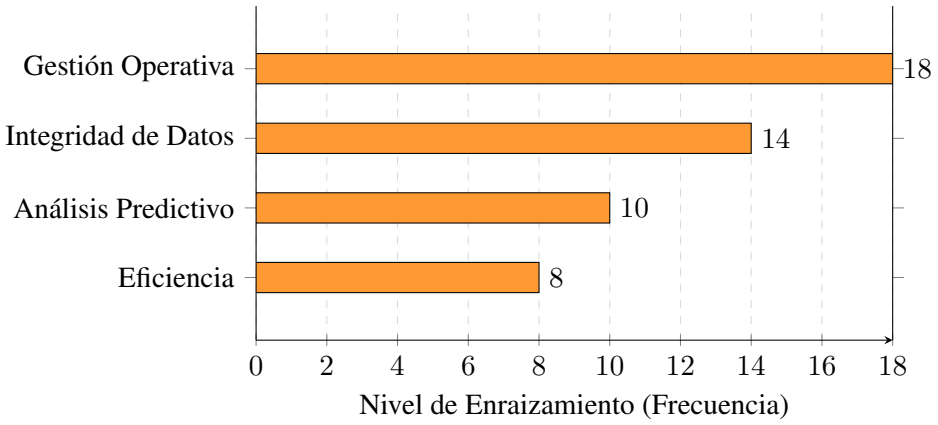


Figura 1: Análisis de categorías emergentes mediante procesamiento de lenguaje natural.

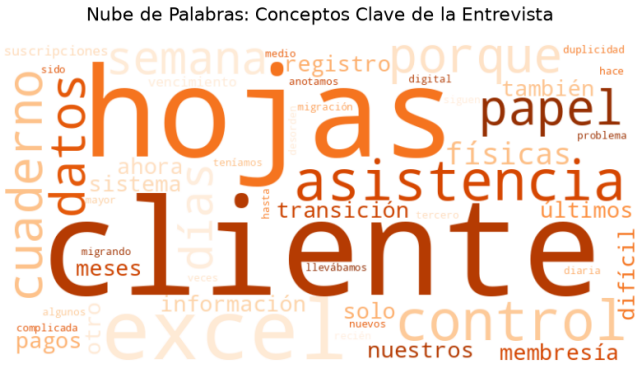


Figura 2: Nube de palabras generada a partir del procesamiento NLP de la entrevista.

Como se observa en las Figuras 1 y 2, las categorías con mayor nivel de enraizamiento fueron: *"Gestión Operativa"*, *"Integridad de Datos"* y *"Análisis Predictivo"*. El análisis de la entrevista revela que la duplicidad de información en registros físicos constituye la principal limitación operativa identificada, ya que impide el seguimiento oportuno de socios y dificulta la detección temprana de patrones de abandono. Estos hallazgos justifican la implementación de un modelo predictivo, ya que sin trazabilidad digital el gimnasio no tiene forma de anticipar cuándo un socio está por abandonar.

- **Encuesta:** Aplicada de forma digital a los socios durante el periodo de marzo 2026, enfocándose en la experiencia de usuario y el seguimiento de su progreso

físico.

Validación de la encuesta: La fiabilidad del instrumento se determinó mediante el coeficiente alfa de Cronbach, calculado sobre una muestra piloto de 10 encuestados antes de aplicarlo a los 132 participantes. Este paso valida la consistencia interna del instrumento previo a su uso definitivo [16].

Tabla 19: Análisis de Fiabilidad (Alfa de Cronbach)

ENCUESTADOS	**Puntuación de Ítems**												**SUMA**
	3	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	
E1	4	5	3	4	5	5	3	4	4	4	4	5	50
E2	5	5	4	4	4	5	4	4	4	4	4	4	50
E3	5	5	5	5	5	5	5	5	5	5	5	5	60
E4	5	5	5	5	4	5	4	5	3	5	5	4	55
E5	2	5	3	5	4	5	2	4	5	4	5	4	48
E6	5	5	5	5	5	5	5	5	5	5	5	5	60
E7	4	5	5	3	5	4	4	5	4	5	5	5	54
E8	5	5	5	5	5	5	5	5	5	5	5	5	60
E9	5	5	5	5	4	5	4	5	3	5	5	4	55
E10	2	5	3	5	4	5	2	4	5	4	5	4	48
VARIANZA	1,36	0,00	0,81	0,44	0,25	0,09	1,21	0,24	0,61	0,24	0,16	0,25	–

Cálculo	**Valor**
SUMATORIA DE VARIANZAS ($\sum S_i^2$)	5,660
VARIANZA DE LA SUMA DE LOS ÍTEMS (S_t^2)	21,400

$\boldsymbol{\alpha}$ **:**	Coeficiente de confiabilidad del cuestionario	**0,802**
k :	Número de ítems del instrumento	12
$\sum S_i^2$:	Sumatoria de las varianzas de los ítems.	5,660
S_t^2 :	Varianza total del instrumento.	21,400

Como se muestra en la Tabla 19, las preguntas de satisfacción presentan una alta consistencia interna en la escala de Likert utilizada, con un valor de 0.802, lo que indica que el instrumento es altamente confiable para los fines de la investigación.

Pregunta 1: ¿Cuál es su rango de edad?

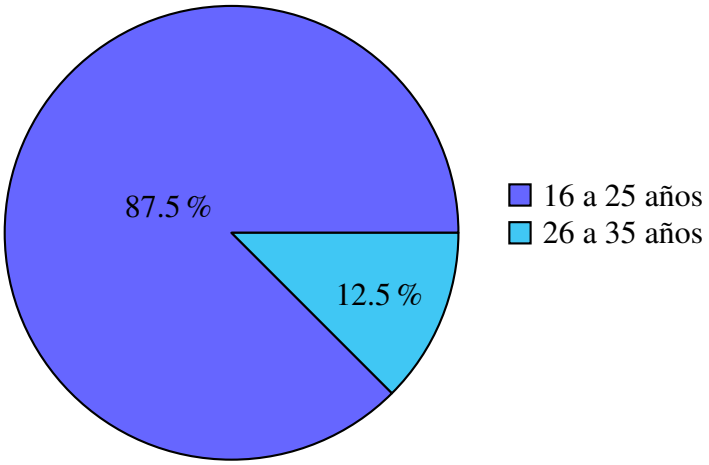


Figura 3: Distribución porcentual de la Pregunta 1

Tabla 20: Distribución por rango de edad de los socios

Categoría	**Frecuencia**	**Porcentaje (%)**
16 a 25 años	116	87.5 %
26 a 35 años	16	12.5 %
Total	**132**	**100.0 %**

Análisis: El 87.5 % de los socios pertenece al rango de 16 a 25 años (Tabla 20). Este perfil joven implica mayor variabilidad en la continuidad de membresías, dado que los adultos en esa franja son más susceptibles a cambios de rutina y prioridades personales.

Pregunta 2: ¿Cuánto tiempo lleva siendo miembro del gimnasio?

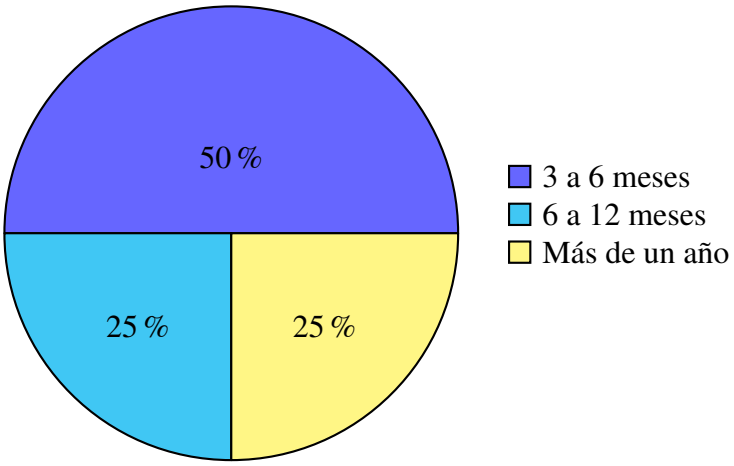


Figura 4: Distribución porcentual del tiempo de permanencia de los socios

Tabla 21: Tiempo de permanencia de los socios

Categoría	Frecuencia	Porcentaje (%)
3 a 6 meses	66	50.0 %
6 a 12 meses	33	25.0 %
Más de un año	33	25.0 %
Total	**132**	**100.0 %**

Análisis: La mitad de los socios encuestados lleva entre 3 y 6 meses en el gimnasio (Tabla 21). Este dato coincide con lo señalado por el administrador en la entrevista (Tabla 18): la etapa inicial es la de mayor riesgo de abandono y, por tanto, la ventana crítica de intervención del modelo predictivo.

Pregunta 3: El registro de entrada al gimnasio es rápido y sencillo.

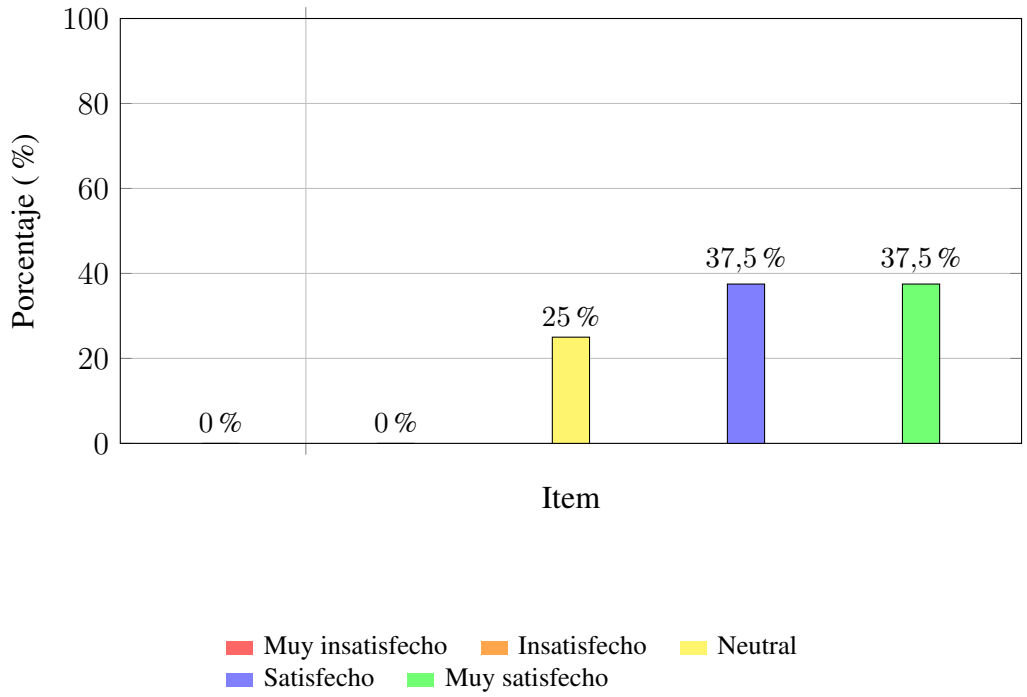


Figura 5: Distribución porcentual de la rapidez en el registro de asistencia

Tabla 22: Rapidez en el registro de entrada

Escala	Frecuencia	Porcentaje (%)
3 (Neutral)	33	25.0 %
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	49	37.5 %
Total	**132**	**100.0 %**

Análisis: El 75 % de los socios valora positivamente la rapidez del registro de entrada (Tabla 22). El 25 % restante mantiene una postura neutral. Los problemas de registros físicos documentados en la entrevista explican ese margen: el proceso de control de acceso tiene oportunidad concreta de mejora.

Pregunta 4: La renovación de mi membresía es un proceso ágil.

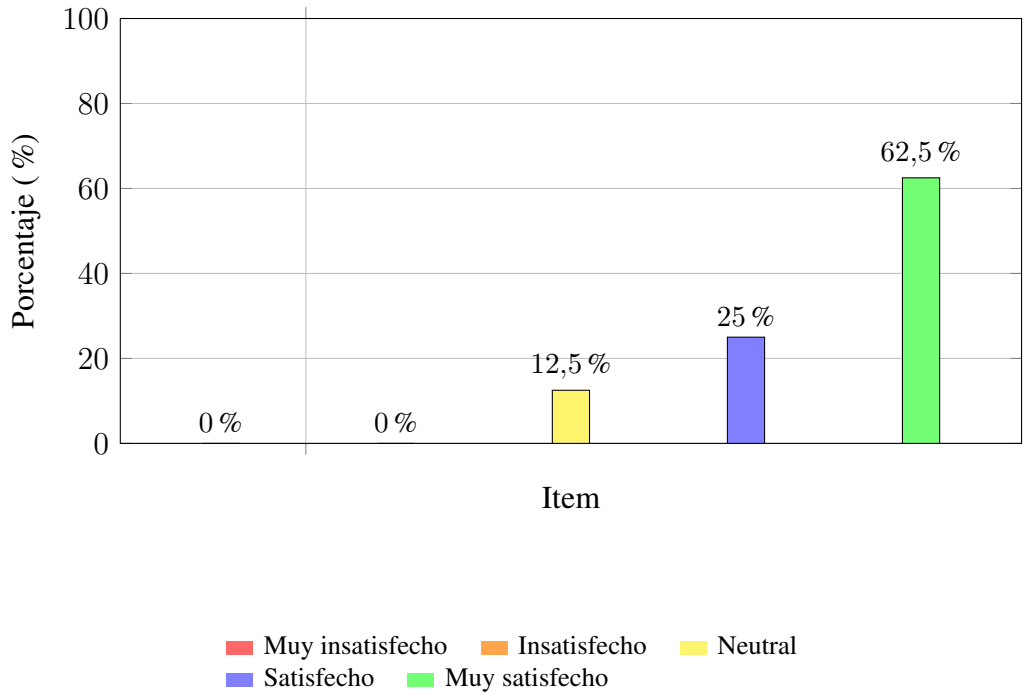


Figura 6: Distribución porcentual de la agilidad en la renovación de membresía

Tabla 23: Agilidad en la renovación de membresía

Escala	Frecuencia	Porcentaje (%)
3 (Neutral)	17	12.5 %
4 (Satisfecho)	33	25 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: El proceso de renovación de membresía satisface al 87.5 % de los socios (Tabla 23). El 12.5 % de neutralidad coincide con la problemática de actualización manual señalada por el administrador (Tabla 18): digitalizar el proceso de renovación es una deuda operativa pendiente.

Pregunta 5: El personal del gimnasio responde de manera clara y amable a mis consultas.

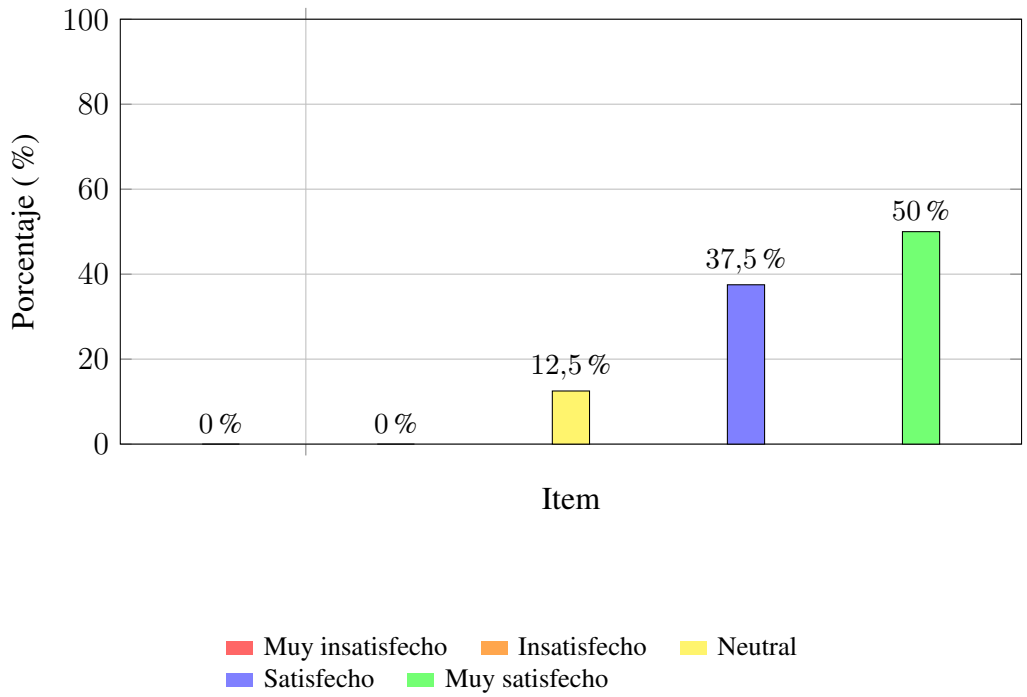


Figura 7: Distribución porcentual de la claridad y amabilidad del personal

Tabla 24: Atención del personal ante consultas

Escala	Frecuencia	Porcentaje (%)
3 (Neutral)	17	12.5 %
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	66	50.0 %
Total	**132**	**100.0 %**

Análisis: La atención del personal recibe valoración positiva del 87.5 % de los socios, con la mitad ubicada en el nivel más alto (Tabla 24). El 12.5 % de neutralidad indica que la comunicación del personal tiene espacio para afinarse. No implica insatisfacción.

Pregunta 6: La información sobre los planes de membresía y beneficios es fácil de entender.

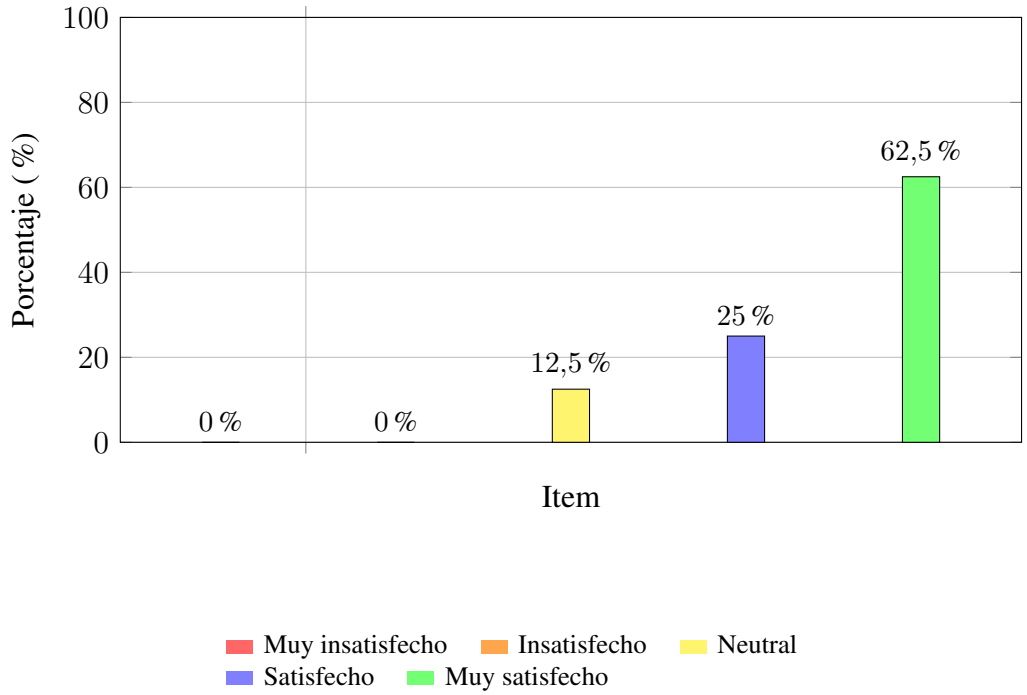


Figura 8: Distribución porcentual de la claridad en la información de planes y beneficios

Tabla 25: Claridad en la información de membresías

Escala	Frecuencia	Porcentaje (%)
3 (Neutral)	17	12.5 %
4 (Satisfecho)	33	25.0 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: El 87.5 % de los socios encuentra clara y comprensible la información sobre planes y beneficios (Tabla 25). El 12.5 % de neutralidad señala que la presentación de la oferta puede refinarse, sin que eso afecte la satisfacción general con el servicio.

Pregunta 7: Las instalaciones y equipos del gimnasio están en buen estado.

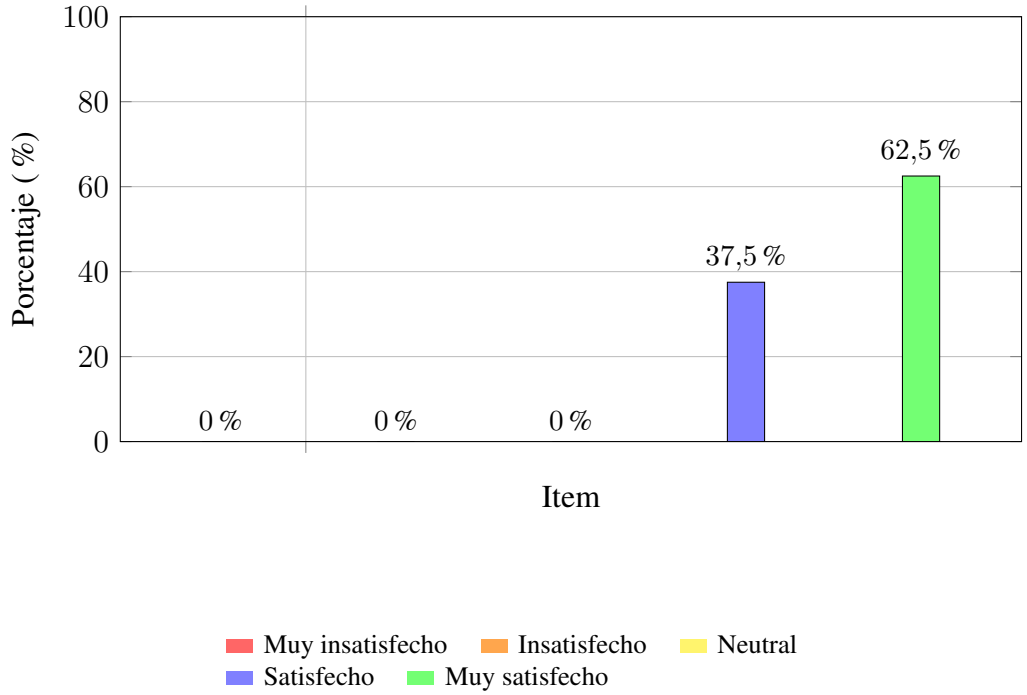


Figura 9: Distribución porcentual del estado de instalaciones y equipos

Tabla 26: Estado de instalaciones y equipos

Escala	Frecuencia	Porcentaje (%)
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: Las instalaciones y equipos obtienen aprobación unánime: el 62.5 % los califica en el nivel máximo y el 37.5 % restante en el nivel inmediato inferior (Tabla 26). El espacio físico no representa un factor de riesgo para la retención.

Pregunta 8: El horario de atención del gimnasio es conveniente para mí.

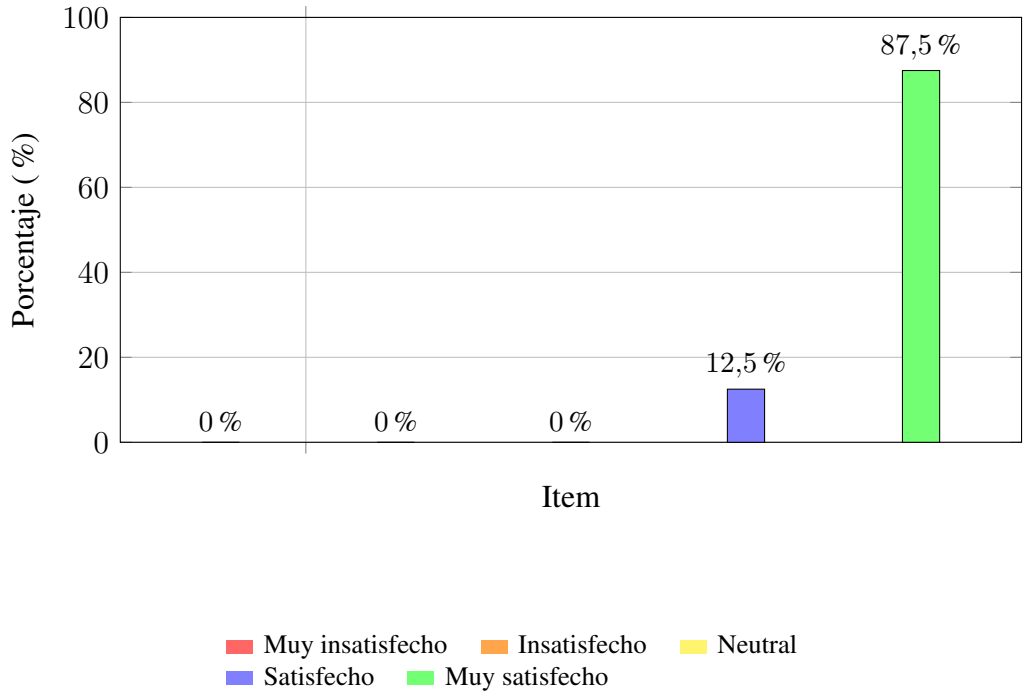


Figura 10: Distribución porcentual de la conveniencia del horario de atención

Tabla 27: Horario de atención

Escala	**Frecuencia**	**Porcentaje (%)**
4 (Satisfecho)	17	12.5 %
5 (Muy satisfecho)	115	87.5 %
Total	**132**	**100.0 %**

Análisis: El horario de atención registra el mayor consenso positivo del instrumento: el 87.5 % de los socios lo ubica en el nivel máximo y no se registra ninguna respuesta neutra o negativa (Tabla 27). Los turnos actuales se adecuan a las necesidades de la base de socios.

Pregunta 9: Siento que el gimnasio realiza un buen seguimiento de mi progreso físico.

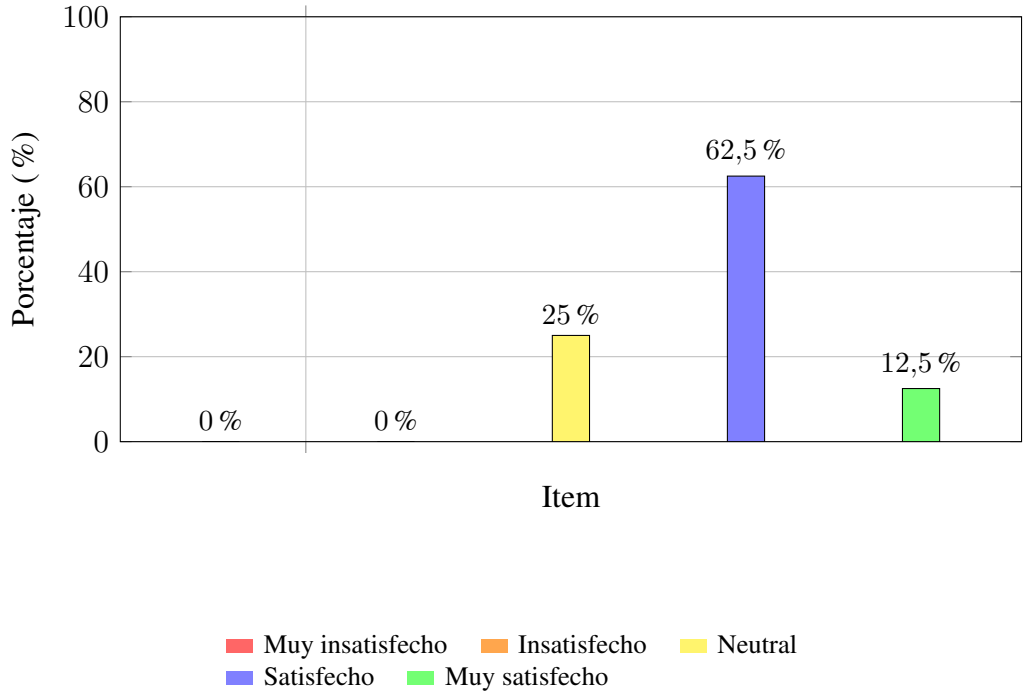


Figura 11: Distribución porcentual del seguimiento del progreso físico

Tabla 28: Seguimiento del progreso físico

Escala	**Frecuencia**	**Porcentaje (%)**
3 (Neutral)	33	25.0 %
4 (Satisfecho)	82	62.5 %
5 (Muy satisfecho)	17	12.5 %
Total	**132**	**100.0 %**

Análisis: El seguimiento del progreso físico es el indicador con la menor proporción en el nivel máximo: solo el 12.5 % de los socios lo alcanza (Tabla 28). El gimnasio carece de herramientas sistemáticas de retroalimentación física. Esa brecha justifica directamente el dashboard predictivo de Strong 2.0, diseñado para ofrecer al administrador datos objetivos sobre la evolución de cada socio.

Pregunta 10: Los beneficios que recibo valen lo que pago por la membresía.

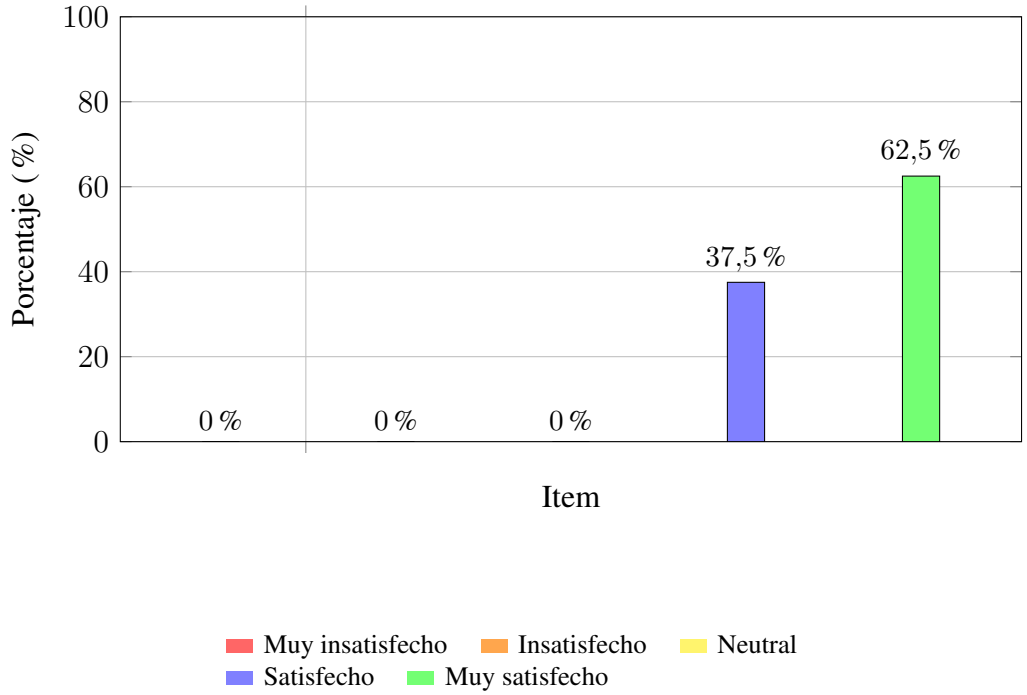


Figura 12: Distribución porcentual de la relación beneficio-precio de la membresía

Tabla 29: Relación precio - calidad

Escala	**Frecuencia**	**Porcentaje (%)**
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: La relación beneficio-precio obtiene evaluación positiva sin excepción: el 62.5 % la ubica en el nivel máximo y el 37.5 % restante en el nivel inmediato inferior (Tabla 29). La propuesta de valor del gimnasio no genera fricción económica en la decisión de renovar.

Pregunta 11: El ambiente del gimnasio es motivador y agradable.

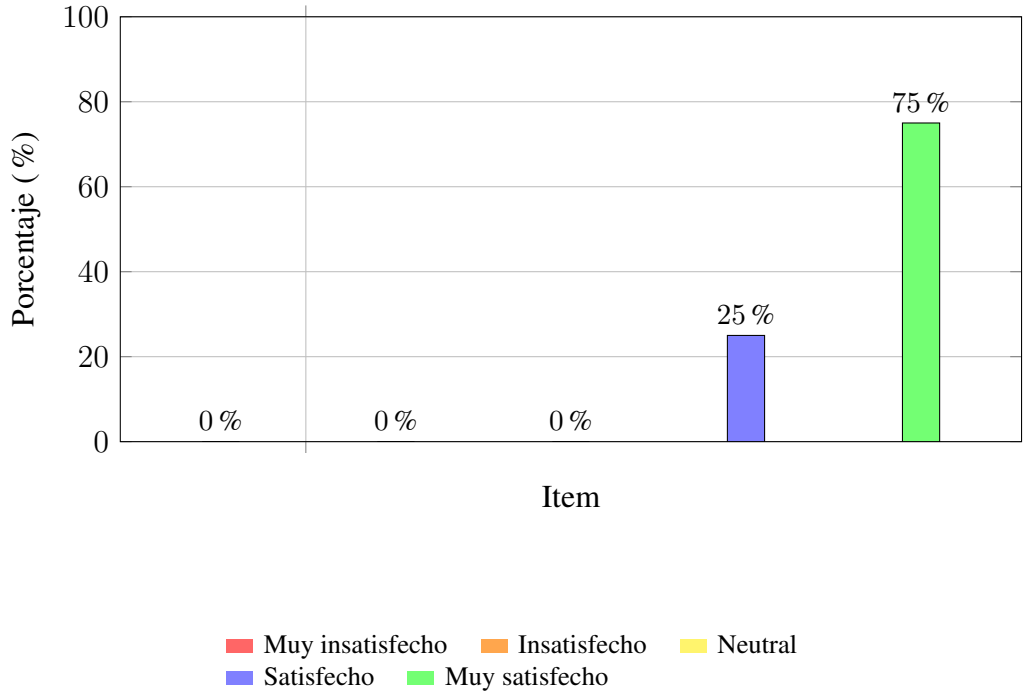


Figura 13: Distribución porcentual del ambiente motivador del gimnasio

Tabla 30: Ambiente del gimnasio

Escala	**Frecuencia**	**Porcentaje (%)**
4 (Satisfecho)	33	25.0 %
5 (Muy satisfecho)	99	75.0 %
Total	**132**	**100.0 %**

Análisis: El 75 % de los socios califica el ambiente del gimnasio en el nivel máximo de satisfacción (Tabla 30). Ningún encuestado reportó percepción neutra o negativa. Un clima motivador sostenido actúa como factor protector frente al abandono.

Pregunta 12: Los entrenadores están disponibles y me guían de manera efectiva.

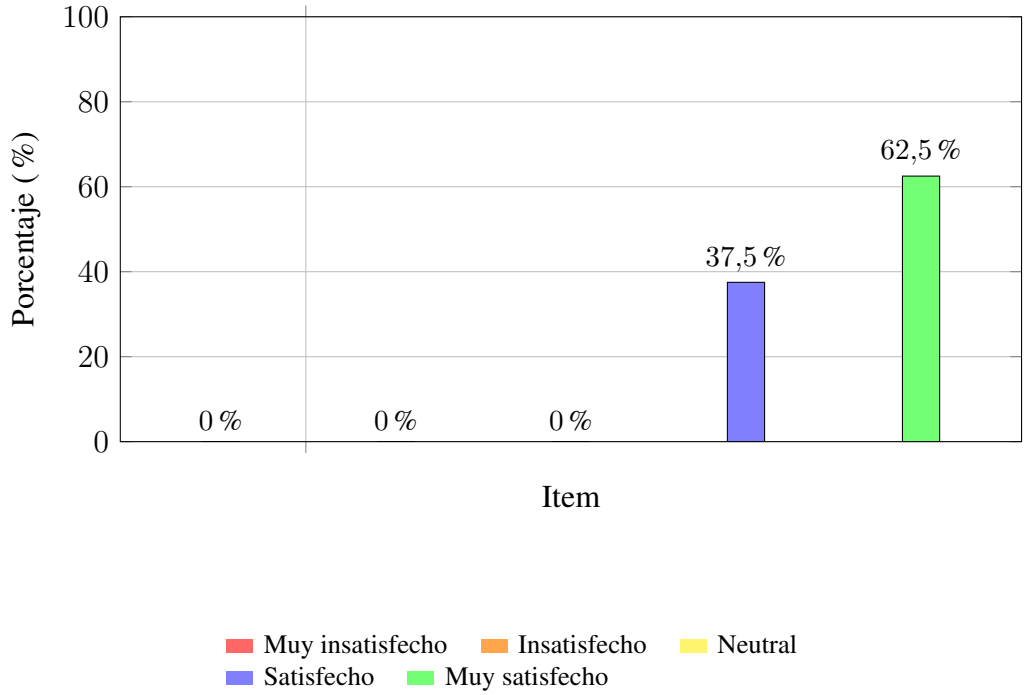


Figura 14: Distribución porcentual de la disponibilidad y efectividad de los entrenadores

Tabla 31: Guía de los entrenadores

Escala	**Frecuencia**	**Porcentaje (%)**
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: Los entrenadores reciben valoración positiva del total de socios encuestados, con un 62.5 % en el nivel más alto (Tabla 31). La calidad del acompañamiento es un activo que el sistema puede potenciar al dirigir la atención del entrenador hacia los socios identificados en riesgo por el modelo.

Pregunta 13: En general, ¿cómo calificaría su experiencia en el gimnasio?

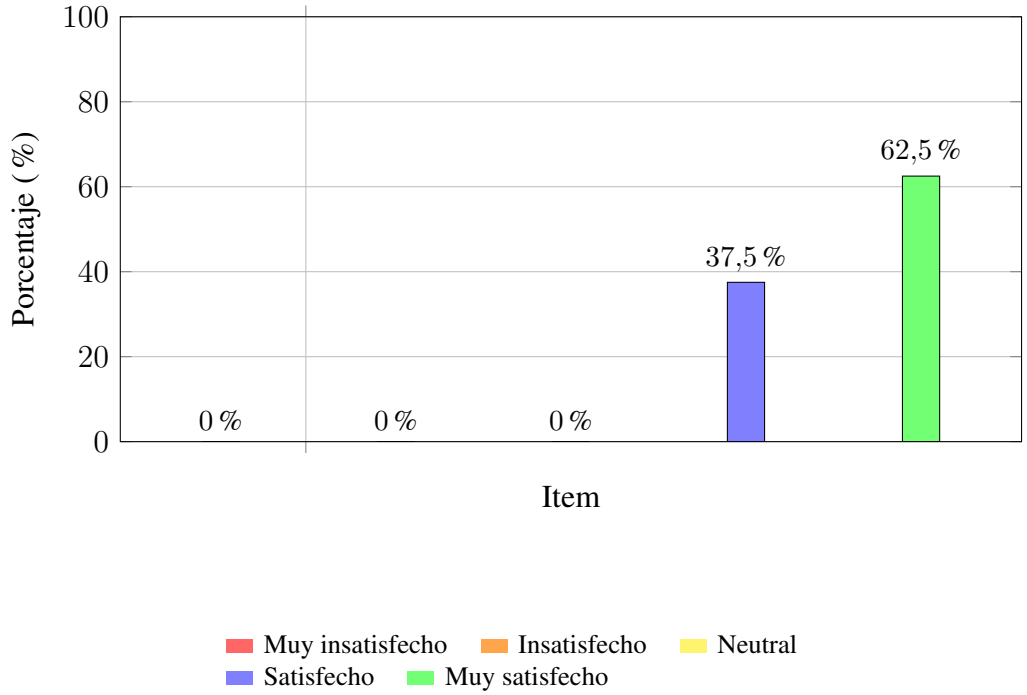


Figura 15: Distribución porcentual de la experiencia general en el gimnasio

Tabla 32: Calificación general de la experiencia

Escala	**Frecuencia**	**Porcentaje (%)**
4 (Satisfecho)	50	37.5 %
5 (Muy satisfecho)	82	62.5 %
Total	**132**	**100.0 %**

Análisis: La experiencia general es el único indicador sin respuestas neutras ni negativas entre los trece evaluados (Tabla 32). El servicio cubre las expectativas del usuario en términos agregados. Esta base de satisfacción es el punto de partida sobre el cual el modelo predictivo opera para anticipar el abandono antes de que ocurra.

2.2.4 Procesamiento y análisis de datos

El análisis de la información recolectada se orienta a identificar las necesidades de gestión del personal administrativo del gimnasio Strong 2.0. Para el tratamiento textual de la entrevista semiestructurada se implementó una cadena de procesamiento de lenguaje natural (PLN) en Python, compuesta por cuatro etapas secuenciales: (1) *preprocesamiento del texto*: conversión a minúsculas y eliminación de signos de puntuación mediante expresiones regulares, seguida de tokenización por espacios; (2) *filtrado de términos vacíos*: aplicación de una lista de palabras funcionales del español adaptada al dominio, es decir, determinantes, preposiciones, conjunciones y verbos auxiliares de alta frecuencia, conservando únicamente los términos con carga semántica; (3) *análisis de frecuencia de términos*: conteo de ocurrencias de cada término en el corpus filtrado; y (4) *codificación por categorías emergentes*: clasificación de los términos en cuatro categorías temáticas: Gestión Operativa, Integridad de Datos, Eficiencia/Alertas y Análisis Predictivo, con cálculo del nivel de *enraizamiento* de cada categoría como la suma de frecuencias de sus términos asociados.

La validez de este enfoque se sustenta en tres argumentos. Primero, la metodología reproduce de forma programática el análisis de contenido cualitativo realizado en herramientas como Atlas.ti, donde el *enraizamiento* (groundedness) cuantifica la presencia de un código en el corpus y constituye una métrica consolidada en la investigación cualitativa. Segundo, las categorías se construyeron de forma deductiva a partir del marco teórico y del problema identificado, lo que asegura coherencia con los objetivos de la investigación y evita la generación de categorías arbitrarias. Tercero, la implementación en Python hace que el proceso sea reproducible y auditable: las mismas entradas producen siempre los mismos resultados, lo cual no está garantizado en una codificación manual. Para el volumen de información recolectada en esta investigación, compuesto por una entrevista semiestructurada de catorce preguntas y un cuestionario de trece ítems con 132 respondentes, este nivel de automatización es adecuado y proporcional a la naturaleza mixta del estudio.

El Código 1 presenta la implementación de las etapas 1 a 4 del proceso descrito.

```
import re
from collections import Counter

# Etapa 1 y 2: preprocesamiento y filtrado de terminos vacios
def preprocess_text(texto):
    clean_text = re.sub(r'[^\w\s]', '', texto.lower())
    palabras = clean_text.split()
    stopwords = {
        'de', 'la', 'en', 'el', 'y', 'los', 'que', 'un', 'una', 'para',
        'con', 'es', 'del', 'las', 'por', 'lo', 'se', 'su', 'al', 'o',
        'este', 'esta', 'alex', 'objetivo', 'identifica', 'identificar',
        'proceso', 'como', 'pero', 'mas', 'sus', 'todo', 'estamos',
        'tenemos', 'donde', 'cada', 'estos', 'esta', 'estan'
    }
    return [w for w in palabras if w not in stopwords and len(w) > 3]

palabras_filtradas = preprocess_text(corpus_entrevista)

# Etapa 3: frecuencia de terminos
frecuencias = Counter(palabras_filtradas)

# Etapa 4: codificacion por categorias emergentes (enraizamiento)
categorias = {
    "Gestion Operativa": ["asistencia", "suscripciones", "registro",
                          "clientes", "pagos"],
```

```
    "Integridad de Datos": ["duplicidad", "excel", "hojas", "papel",
                            "inconsistencias", "migracion"],
    "Eficiencia/Alertas": ["alertas", "vencimientos", "tiempo",
                           "automaticas", "urgente"],
    "Analisis Predictivo": ["patrones", "comportamiento", "cancelar",
                            "desercion", "frecuencia"]
}

conteo = []
for categoria, terminos in categorias.items():
    for termino in terminos:
        frecuencia = palabras_filtradas.count(termino)
        if frecuencia > 0:
            conteo.append({
                "Categoria": categoria,
                "Codigo": termino,
                "Enraizamiento": frecuencia
            })
```

Código 1: Implementación del proceso PLN aplicado a la entrevista semiestructurada.

Según la entrevista realizada al administrador Alex Flores, en la Pregunta 2 se observa una crisis de integridad de datos derivada de una transición desordenada entre registros físicos y archivos de Excel. El hallazgo más alarmante es la pérdida de 18 meses de historial operativo, contando actualmente con solo 15 días de datos digitalizados, como se detalla en la Pregunta 3. Para el administrador, esta situación genera una vulnerabilidad operativa que se traduce en aproximadamente 15 inconsistencias semanales en el registro de pagos de los 200 socios activos, un aspecto detallado en la Pregunta 2, validando la necesidad de una base de datos confiable que centralice la información y elimine la duplicidad de procesos.

Según la entrevista al administrador, el control manual de asistencia y vencimientos de membresías consume entre 2 a 3 horas diarias de la jornada laboral del personal. Esta carga motivó el desarrollo del módulo de automatización de alertas en Strong 2.0.

Las preguntas 1 y 2 revelan que el 87.5 % de los socios pertenece a un segmento joven y el 75 % lleva menos de un año en el gimnasio. Esta combinación define una base de clientes con alta disposición hacia procesos digitales.

Las preguntas 3, 4, 5 y 6 muestran satisfacción mayoritaria con los procesos operativos. El 25 % de neutralidad en el registro de acceso confirma que centralizar la gestión en una base de datos única sigue siendo una necesidad pendiente.

Las preguntas 7, 8, 11 y 12 registran alta satisfacción en el entorno físico y el horario del gimnasio. Estos aspectos no representan factores críticos de deserción.

Las preguntas 9, 10 y 13 revelan el hallazgo central del instrumento: el seguimiento del progreso físico registra el menor nivel de satisfacción, con solo el 12.5 % en el nivel máximo. La ausencia de herramientas de retroalimentación física fundamenta la necesidad del modelo predictivo.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

Este capítulo presenta los resultados obtenidos en cada etapa del proyecto. Se inicia con el análisis de los procesos manuales del gimnasio Strong 2.0 y la identificación de requisitos funcionales. A continuación, se expone la evaluación comparativa de algoritmos de aprendizaje automático y la selección del modelo predictivo. Luego se detalla la implementación del sistema web con el modelo integrado. El capítulo concluye con una síntesis de resultados cuantitativos y una discusión que contrasta los hallazgos con los antecedentes revisados.

3.1 Identificar los procesos manuales de gestión de suscripciones y control de asistencia para establecer los requisitos funcionales del nuevo sistema

Se analizaron los procesos administrativos actuales del gimnasio Strong 2.0 con el fin de identificar los requisitos funcionales del sistema.

Análisis de la situación actual y recolección de información

Para conocer la realidad del negocio, se realizaron una entrevista semiestructurada al administrador y una encuesta a los socios activos. Los resultados de la entrevista se presentan en la Tabla 18; los de la encuesta, en las tablas correspondientes a las preguntas 1 a 13 de la subsección anterior. La información obtenida en la entrevista fue procesada mediante la cadena de procesamiento de lenguaje natural (PLN) en Python descrita en la sección de metodología, que realizó el preprocesamiento del texto, el filtrado de términos vacíos, el conteo de frecuencias y la codificación en cuatro categorías temáticas con su respectivo nivel de enraizamiento. Este procedimiento permitió cuantificar el peso relativo de cada categoría y determinar cuáles concentran la mayor densidad de referencias en el discurso del administrador.

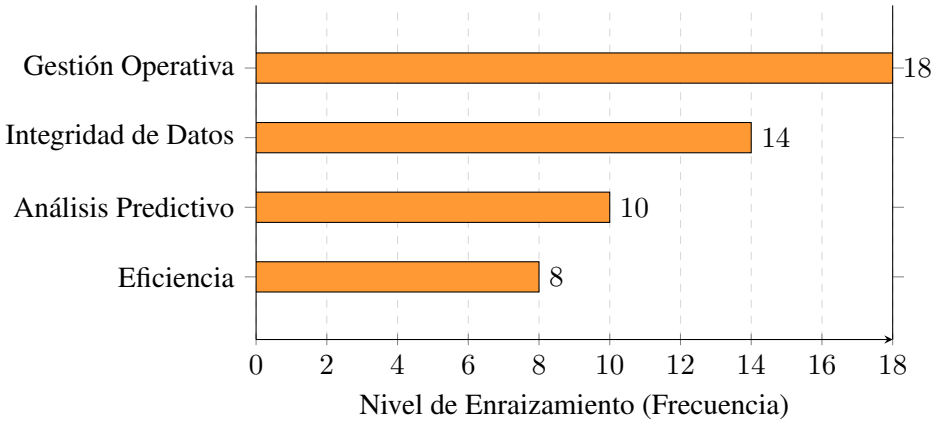


Figura 16: Análisis de categorías emergentes mediante procesamiento de lenguaje natural.

La Figura 16 indica que la gestión operativa y el orden de los datos concentran el mayor peso en la problemática identificada. El uso de métodos manuales genera las siguientes limitaciones operativas:

- Los registros de entrada y el control de pagos se manejan de forma manual, sin un sistema centralizado.
- No existe una forma real de saber por qué un socio deja de ir; todo queda en simples suposiciones.

Modelado del proceso actual

Para entender las ineficiencias del gimnasio, se mapearon cuatro procesos administrativos clave mediante diagramas BPMN: registro de socios, control de asistencia, seguimiento de membresías y cobro de mensualidades.

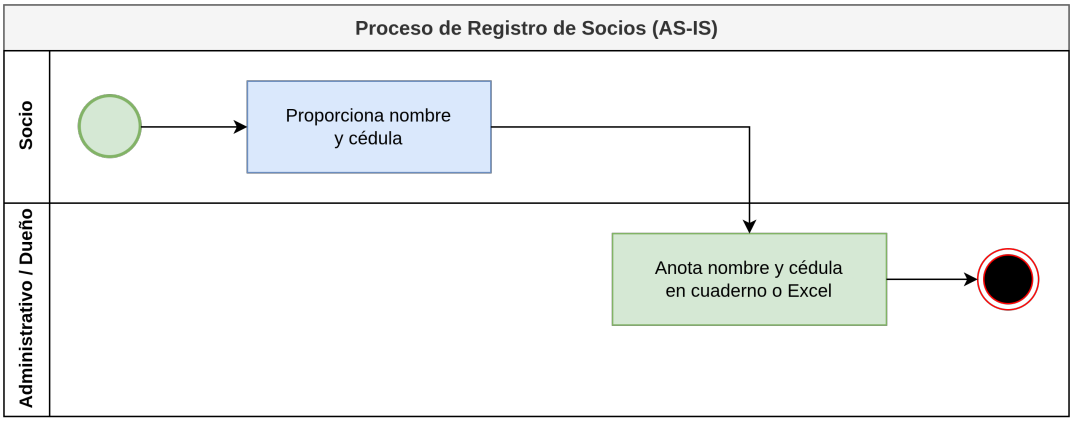


Figura 17: BPMN: Proceso de registro de socios (proceso actual)

El proceso de registro de socios (Figura 17) se limita al ingreso de datos básicos de identificación: nombre y número de cédula, sin persistencia digital estructurada que permita su consulta posterior.

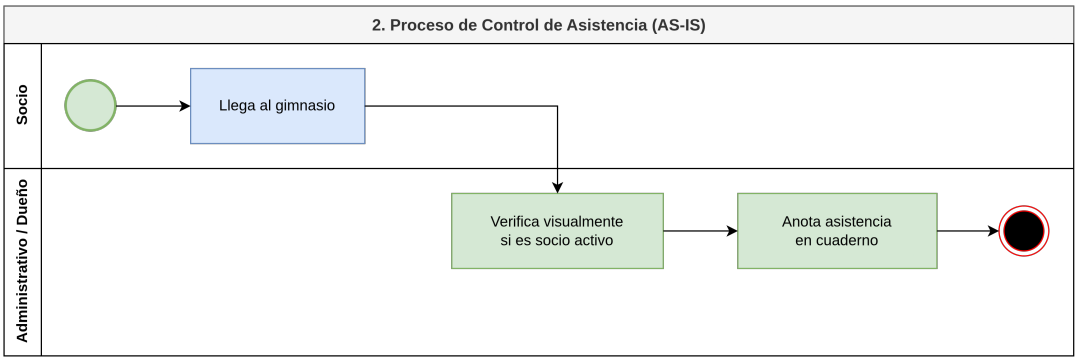


Figura 18: BPMN: Proceso de control de asistencia (proceso actual)

El proceso de control de asistencia (Figura 18) se ejecuta de forma manual: el personal identifica al socio visualmente y registra su ingreso en un cuaderno, sin ningún mecanismo para detectar inasistencias prolongadas.

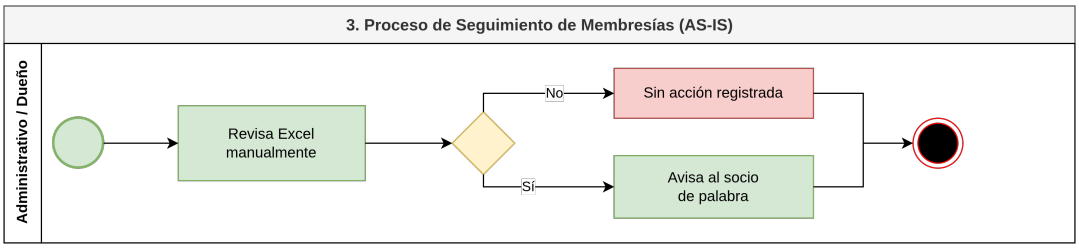


Figura 19: BPMN: Proceso de seguimiento de membresías (proceso actual)

El seguimiento de membresías (Figura 19) carece de automatización: la revisión del estado de cada membresía depende de una verificación manual periódica en una hoja de cálculo. La notificación al socio ocurre únicamente tras el vencimiento y de forma verbal, sin registro de esa comunicación.

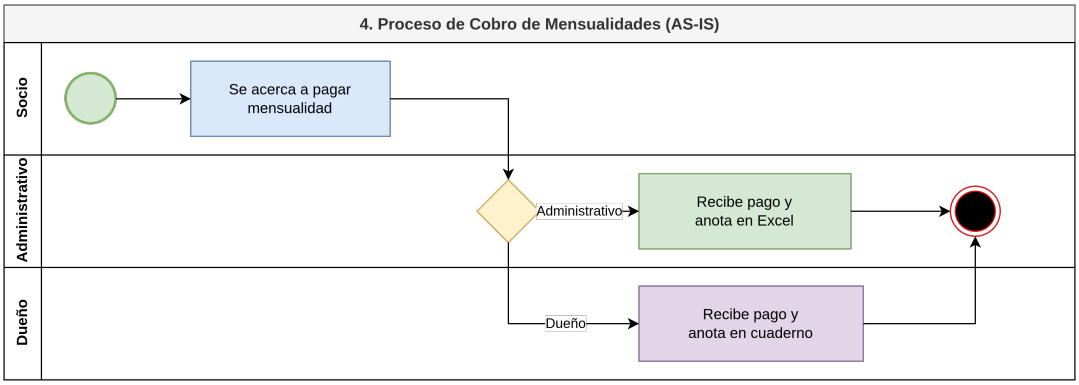


Figura 20: BPMN: Proceso de cobro de mensualidades (proceso actual)

El proceso de cobro de mensualidades (Figura 20) no cuenta con un canal unificado de registro: los pagos se documentan en distintos soportes según el personal que atiende, ya sea hoja de cálculo o cuaderno físico, lo que genera un historial fragmentado e inconsistente.

Los cuatro procesos comparten una característica común: la ausencia de trazabilidad digital sistemática. Sin registros estructurados, no es posible identificar oportunamente patrones de comportamiento asociados al abandono.

Matriz de riesgos y oportunidades (FODA)

Se elaboró una matriz para entender cuál es el estado actual del gimnasio y qué puede mejorar con el nuevo sistema.

Tabla 33: Matriz FODA de la gestión administrativa en Strong 2.0.

Fortalezas	**Oportunidades**
Base de datos de 200 socios activos y disposición tecnológica del segmento joven (87.5 % bajo los 25 años).	Digitalización para mejorar la retención y ofrecer servicios personalizados basados en IA.
Debilidades	**Amenazas**
Pérdida de historial operativo (solo 15 días digitales) y carga administrativa manual de 3 horas diarias.	Incremento de la tasa de deserción por falta de seguimiento y competencia con centros automatizados.

Requerimientos funcionales y no funcionales (RF y RNF)

El diagnóstico de las deficiencias en el manejo manual permitió definir los requisitos funcionales del sistema. Esta estructuración técnica asegura la calidad del software y establece parámetros claros para su validación final.

Requerimientos funcionales (RF): Estas tareas indican cómo el sistema transforma la operación diaria del gimnasio. La Tabla 34 detalla los pasos necesarios para pasar de un modelo antiguo a uno digital y preventivo.

Tabla 34: Lista de requerimientos funcionales.

ID	Nombre	Descripción Técnica
RF-01	Gestión de socios	Guarda y maneja los perfiles para alimentar al modelo de predicción.
RF-02	Control de membresías	Maneja de forma digital las fechas de vencimiento y los pagos.
RF-03	Registro de asistencia	Anota las entradas diarias para entender cómo se comportan los socios.
RF-04	Motor de predicción	Usa algoritmos para calcular qué tan probable es que un socio se vaya.
RF-05	Alertas de riesgo	Envía avisos automáticos según los límites de probabilidad elegidos.
RF-06	Explicación de la IA	Traduce las predicciones para entender qué factores causan el riesgo.

Requerimientos no funcionales (RNF): Estas condiciones aseguran que el sistema sea seguro, mantenible y analíticamente preciso. La Tabla 35 detalla las reglas que guían el comportamiento general de la plataforma.

Tabla 35: Lista de requerimientos no funcionales del sistema.

ID	Dimensión / Atributo	Especificación Técnica
RNF-01	Seguridad e integridad	El acceso a la información se controla mediante roles. Las credenciales se protegen con hashing irreversible y los datos personales de los socios se enmascaran según el rol del solicitante.
RNF-02	Mantenibilidad y estructura	La arquitectura desacoplada debe permitir actualizar o cambiar el modelo `.joblib` sin alterar el diseño del frontend.
RNF-03	Compatibilidad de datos	El frontend y el backend deben comunicarse sin problemas utilizando el formato estándar JSON para el intercambio de datos.
RNF-04	Exactitud analítica	El modelo predictivo debe alcanzar una exactitud mínima del 80 % y un ROC AUC superior a 0,90, umbrales que garantizan la fiabilidad necesaria para identificar socios en riesgo de deserción.

3.2 Determinar el modelo predictivo más adecuado para predecir la deserción de clientes

La identificación de socios en riesgo de abandono requiere información digital estructurada que los procesos manuales no generan. Para resolver este problema, se evaluaron seis algoritmos de aprendizaje automático supervisado sobre el conjunto de datos del gimnasio, con el objetivo de seleccionar el modelo con mayor capacidad predictiva e integrarlo al sistema Strong 2.0.

El problema se formuló como una clasificación binaria: el modelo asigna a cada socio la clase de riesgo de deserción (1) o la clase activa (0). Para medir el desempeño de cada algoritmo se utilizó la matriz de confusión, que descompone los resultados en cuatro categorías. Un Verdadero Positivo (VP) ocurre cuando el modelo detecta correctamente a un socio en riesgo, lo que permite al gimnasio intervenir a tiempo.

Un Verdadero Negativo (VN) confirma que el socio permanecerá activo y no requiere acción. Un Falso Positivo (FP) genera una alerta innecesaria sobre alguien que no iba a irse. El Falso Negativo (FN) es el error más costoso: el modelo no detecta a un socio que sí va a desertar y el gimnasio lo pierde sin poder actuar.

La variable objetivo se definió a partir del umbral operativo registrado por el gimnasio: un socio se clasifica en riesgo cuando acumula tres semanas consecutivas de inasistencia. El modelo predice si un socio alcanzará ese umbral, permitiendo al personal intervenir antes de que se produzca el abandono.

Métricas de evaluación

Se calcularon cinco métricas para comparar los algoritmos. La prioridad fue la **Sensibilidad**, dado que el costo de no detectar a un socio que sí va a desertar supera al de una alerta innecesaria:

- **Exactitud (Accuracy):** Indica el porcentaje total de predicciones correctas sobre el total de casos evaluados.

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN}$$

- **Sensibilidad (Recall):** Es la métrica más importante en este proyecto. Mide la habilidad del sistema para detectar a los socios que realmente están en riesgo de irse. Un valor alto asegura que se pierdan pocos casos críticos.

$$\text{Recall} = \frac{VP}{VP + FN}$$

- **Precisión:** Mide qué proporción de las predicciones de deserción son realmente correctas.

$$\text{Precisión} = \frac{VP}{VP + FP}$$

- **F1-Score:** Ofrece un balance justo entre la precisión y la sensibilidad. Es muy

útil cuando los grupos de datos no están repartidos de forma igualitaria.

$$\text{F1-Score} = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

- **ROC-AUC:** Evalúa qué tan bien el modelo diferencia entre un socio que se queda y uno que se va en distintos escenarios. Un valor cercano a 1.0 indica una separación perfecta entre clases.

Evaluación comparativa de algoritmos

El proceso partió de dos fuentes complementarias. La primera son los registros del gimnasio Strong 2.0: 150 fichas de socios construidas a partir de dos fuentes físicas: el cuaderno de planes de membresía, donde el administrador anotaba manualmente el plan contratado, las fechas de inicio y vencimiento, y el estado de renovación de cada socio; y las planillas de cálculo de asistencia, que registraban la frecuencia de visitas. La variable objetivo (*deserción*) es binaria: desertor si la membresía venció sin renovarse dentro del período de gracia, activo si el socio renovó. La segunda fuente es un *dataset* público de 4.000 registros obtenido de Kaggle [1], con variables equivalentes a las del sistema, usado para ampliar el volumen de entrenamiento dado que el conjunto local resultaba insuficiente para entrenar un modelo generalizable. La compatibilidad de variables entre ambas fuentes se verificó mediante el mapeo semántico de cada columna del conjunto local con su equivalente en el dataset de Kaggle, tal como muestra la Tabla 36. Ambos conjuntos miden los mismos fenómenos operativos, a saber, antigüedad, frecuencia de asistencia, duración del contrato y edad, bajo nombres distintos, lo que permitió construir un esquema unificado antes del procesamiento.

Tabla 36: Mapeo de variables entre el dataset local (Strong 2.0) y el dataset de Kaggle.

Variable Kaggle	**Equivalente local / cálculo**	**Tipo**
`Lifetime`	Meses entre `Fecha_de_Registro` y `Fecha_Ultima_Visita`	Continua
`Avg_class_frequency_current_month`	`Visitas_Mensuales` del mes en curso	Continua
`Avg_class_frequency_total`	Total de visitas dividido por meses de permanencia	Continua
`Age`	`Edad` del socio en años	Continua
`Month_to_end_contract`	Diferencia en meses entre `Fecha_fin` y la fecha de cálculo	Continua
`Contract_period`	`Tipo_de_Membresia` convertido a meses (Mensual=1, Trimestral=3, Anual=12)	Ordinal
`Churn`	`Abandono` binarizado (Sí=1, No=0)	Binaria

El conjunto local suma 150 registros, el 3.75 % del volumen total empleado para el entrenamiento. Esta cantidad resulta insuficiente para una validación cruzada con $k = 5$ pliegues, ya que cada partición contendría apenas 30 muestras, volumen que no garantiza la estabilidad de las estimaciones en modelos con seis variables predictoras. Para resolver esta limitación, se incorporó el conjunto de Kaggle [1] como fuente de datos complementaria. Su pertinencia se sustenta en tres criterios: corresponde a un gimnasio urbano cuyo perfil de socios es comparable en distribución de edades, tipos de plan y duración de contratos; comparte las mismas variables operativas que el sistema Strong 2.0 registra; y presenta una tasa de deserción del 26.5 %, prácticamente idéntica al 26.0 % observado en el conjunto local. Esta coincidencia en la variable objetivo refuerza la compatibilidad semántica demostrada en la Tabla 36. La integración de ambas fuentes produjo un conjunto de 4.150 registros con condiciones adecuadas para entrenar y comparar los seis algoritmos evaluados.

Una vez unificadas las fuentes, se aplicó un pipeline de preprocesamiento que incluyó imputación de valores faltantes mediante la mediana como medida de precaución, aunque el conjunto no presentaba valores nulos. Las cinco variables continuas (`Lifetime`, `Avg_class_frequency_current_month`, `Avg_class_frequency_total`, `Age` y `Month_to_end_contract`) y la variable ordinal (`Contract_period`) se normalizaron con *StandardScaler*, garanti-

zando condiciones de procesamiento homogéneas entre los seis algoritmos evaluados, dado que Regresión Logística, KNN y SVM son sensibles a la magnitud de las variables. No fue necesaria ninguna codificación adicional de variables categóricas, ya que el conjunto utilizaba representación binaria (0/1) para todas sus variables cualitativas.

El horizonte temporal de dos años del conjunto local es adecuado para el objetivo de predicción. La variable de respuesta (renovación de membresía) se determina al cierre de cada ciclo de plan, que en Strong 2.0 tiene duración de uno a doce meses. Dos años de registro permiten observar entre dos y veinticuatro ciclos completos por socio, aportando suficientes instancias de decisión para entrenar el modelo. La estabilidad del comportamiento de abandono dentro del período se confirma en la coincidencia de tasas de deserción: 26.0 % en el conjunto local frente al 26.5 % del conjunto de Kaggle, lo que descarta tendencias aberrantes asociadas a la duración del historial.

El Código 2 presenta las tres etapas del proceso: selección de variables, preprocesamiento e imputación, y evaluación comparativa mediante validación cruzada estratificada.

```
# Etapa 1: seleccion de variables predictoras
campos_utiles = [
    'Lifetime', 'Avg_class_frequency_current_month',
    'Avg_class_frequency_total', 'Age',
    'Month_to_end_contract', 'Contract_period'
]
X = df[campos_utiles]
y = df['Churn']

# Etapa 2: imputacion por mediana (precaucion ante nulos)
imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(X)

# Etapa 3: normalizacion + entrenamiento + evaluacion
    cruzada
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state
    =42)
```

```
scoring = {'accuracy': 'accuracy', 'f1': 'f1',
           'recall': 'recall', 'precision': 'precision',
           'roc_auc': 'roc_auc'}

for nombre, modelo in modelos.items():
    pipe = make_pipeline(StandardScaler(), modelo)
    cv_res = cross_validate(pipe, X_imputed, y,
                            cv=skf, scoring=scoring)
```

Código 2: Selección de variables, preprocesamiento y benchmarking comparativo de modelos.

Para determinar qué variables incluir, se calculó la importancia por permutación de todas las columnas disponibles en el dataset original de Kaggle sobre un modelo Gradient Boosting optimizado con Optuna, como se muestra en la Figura 21. Las seis columnas con mayor importancia relativa fueron seleccionadas como variables predictoras finales: antigüedad como socio expresada en meses, frecuencia promedio de asistencia histórica, frecuencia promedio de asistencia en el mes en curso, edad del socio, meses restantes del contrato y duración del plan contratado.

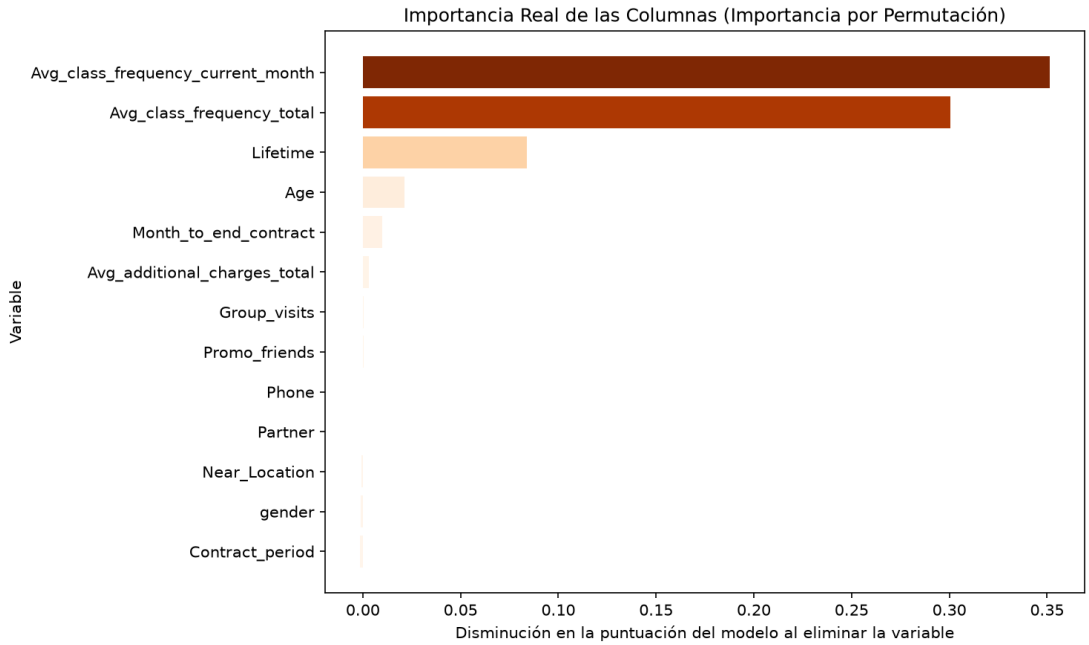


Figura 21: Importancia por permutación de las variables del dataset sobre el modelo Gradient Boosting optimizado con Optuna.

Se evaluaron seis algoritmos bajo las mismas condiciones de procesamiento: Gradient Boosting, SVM, Random Forest, Regresión Logística, KNN y Árbol de Decisión. Todos los modelos se ejecutaron con los parámetros por defecto de scikit-learn, sin ajuste previo de hiperparámetros. Este enfoque permite una comparación equitativa en condiciones base y sigue la práctica habitual de selección de algoritmos: identificar el candidato más prometedor antes de invertir costo computacional en su optimización. Para garantizar la validez estadística, se aplicó validación cruzada estratificada con $k = 5$ pliegues, lo que permitió estimar el rendimiento de cada modelo sobre particiones independientes y reducir el riesgo de sobreajuste. Las métricas de cada pliegue se promediaron para obtener el resultado final.

La métrica prioritaria fue la **Sensibilidad (Recall)**, dado que el costo de un falso negativo, es decir, no detectar a un socio que sí va a desertar, supera con creces el costo de un falso positivo. El **ROC AUC** se utilizó como métrica complementaria para evaluar la capacidad de discriminación global del modelo en distintos umbrales de decisión, mientras que el **F1-Score** permitió verificar el balance entre precisión y sensibilidad ante el desbalance natural de clases del *dataset*. Las curvas ROC comparativas de los seis modelos se presentan en la Figura 22.

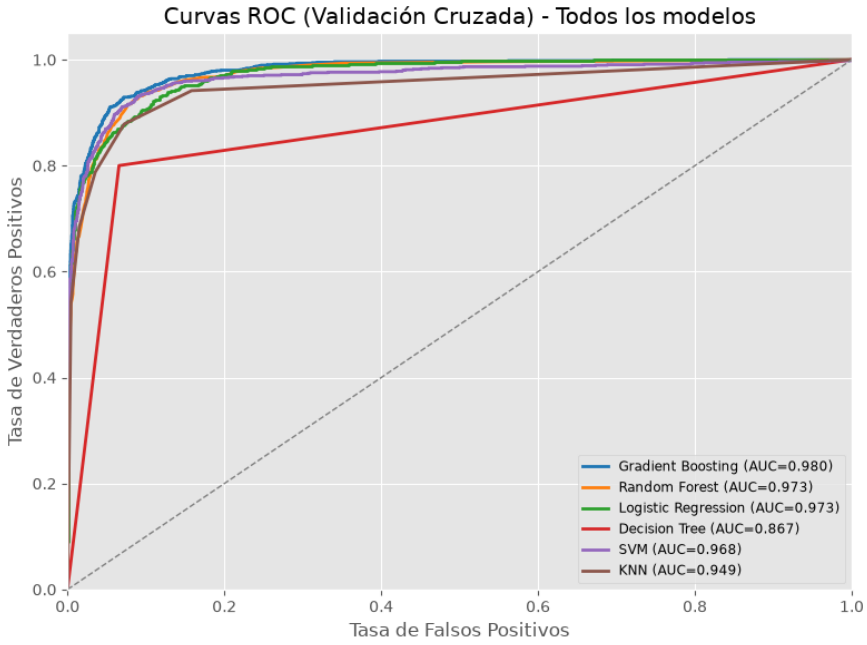


Figura 22: Capacidad de distinción de los modelos (Curvas ROC-AUC).

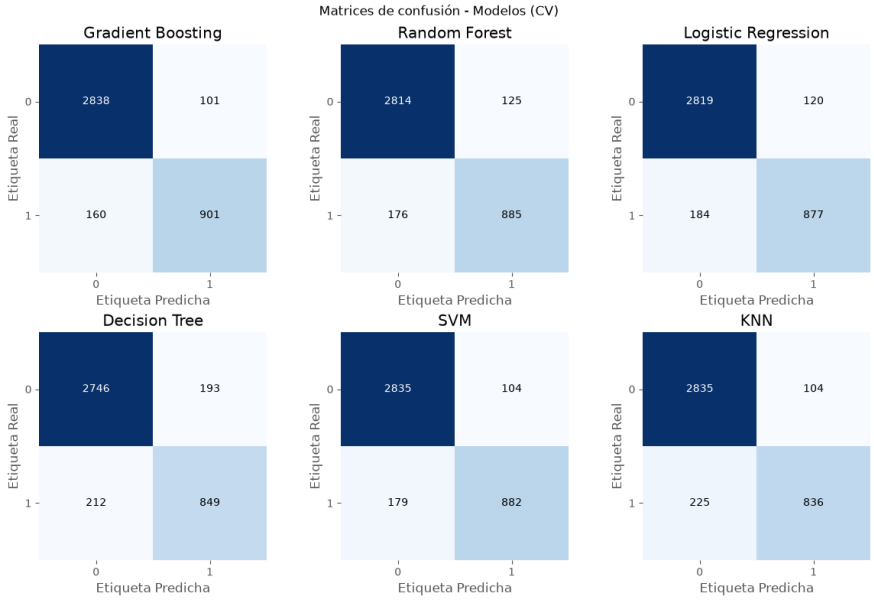


Figura 23: Detalle de aciertos y errores por cada modelo (Matriz de Confusión).

La Tabla 37 resume el desempeño comparativo de los seis modelos evaluados.

Tabla 37: Resultados del análisis comparativo de modelos.

Modelo	**Exactitud**	**F1-Score**	**Sensibilidad**	**Precisión**	**ROC AUC**
Gradient Boosting	**0.9348**	**0.8734**	**0.8492**	**0.8992**	**0.9798**
SVM	0.9293	0.8617	0.8313	0.8947	0.9676
Random Forest	0.9248	0.8547	0.8341	0.8764	0.9725
Regresión Logística	0.9240	0.8521	0.8265	0.8798	0.9729
KNN	0.9178	0.8355	0.7879	0.8893	0.9488
Árbol de Decisión	0.8988	0.8071	0.8001	0.8144	0.8673

El modelo Gradient Boosting alcanzó el mejor desempeño con un 93.48 % de exactitud. Como muestran las Figuras 22 y 23, tanto el análisis de las curvas ROC como el de las matrices de confusión confirman que es la opción más estable. Este algoritmo se eligió como el motor principal de Strong 2.0 por su gran capacidad para manejar datos complejos, como los cambios en la frecuencia de asistencia de los socios.

Una vez seleccionado Gradient Boosting, se procedió a optimizar sus hiperparámetros. Se evaluaron tres alternativas con diferente filosofía de búsqueda: GridSearchCV, Hyperopt y Optuna. Los antecedentes comparativos de cada opción se detallan en la

Tabla 12.

Tabla 38: Matriz de selección de la biblioteca de optimización de hiperparámetros.

Criterio	**Peso**	**GridSearchCV**	**Hyperopt**	**Optuna**
Eficiencia de búsqueda	0.40	1	3	5
Facilidad de uso e integración	0.30	5	3	4
Visualización y seguimiento	0.20	1	2	5
Soporte activo y documentación	0.10	4	3	5
Total		**2.50**	**2.90**	**4.70**

Optuna obtuvo el mayor puntaje total (Tabla 38). GridSearchCV supera en facilidad de configuración al ser nativo de scikit-learn y no requerir instalación adicional; Optuna, en cambio, exige definir una función objetivo y un objeto de estudio. Sin embargo, Optuna se impone en los criterios de mayor peso: su motor de optimización bayesiana aprende de cada evaluación para dirigir la búsqueda hacia las configuraciones más prometedoras, reduciendo el número de iteraciones frente al recorrido exhaustivo de GridSearchCV. Su interfaz de visualización y seguimiento en tiempo real ofrece además una capacidad de inspección del proceso de optimización que los otros candidatos no proporcionan.

```
def objective(trial):
    param = {
        'n_estimators': trial.suggest_int('n_estimators',
            50, 300),
        'learning_rate': trial.suggest_float('learning_rate'
            ,
                                             0.01, 0.3, log=
                                                True),
        'max_depth': trial.suggest_int('max_depth', 3, 8),
        'subsample': trial.suggest_float('subsample', 0.7,
            1.0),
        'random_state': 42
    }
    pipe = Pipeline([
```

```
11          ('imputer', SimpleImputer(strategy='median')),
12          ('clf', GradientBoostingClassifier(**param))
13      ])
14      cv = StratifiedKFold(n_splits=5, shuffle=True,
            random_state=42)
15      score = cross_val_score(pipe, X_train, y_train,
16                              cv=cv, scoring='roc_auc', n_jobs
                                   =-1)
17      return score.mean()
18
19  study = optuna.create_study(direction='maximize')
20  study.optimize(objective, n_trials=30)
```

Código 3: Definición del estudio de optimización bayesiana con Optuna (camposrelevantes.py).

Tabla 39: Hiperparámetros finales del modelo Gradient Boosting obtenidos con Optuna.

Hiperparámetro	**Valor**
`n_estimators`	255
`learning_rate`	0.1649
`max_depth`	3
`subsample`	0.9114

3.3 Implementar el sistema web integrando el modelo predictivo seleccionado para la toma de decisiones

Esta subsección describe la implementación del sistema web Strong 2.0, que integra los módulos de gestión administrativa con el modelo predictivo para apoyar la toma de decisiones basada en datos.

Casos de uso del sistema

La Figura 24 presenta los actores del sistema y los casos de uso agrupados por módulo. El Administrador tiene acceso a todos los módulos, incluyendo la ejecución del motor de inferencia y el registro de asistencia. El Entrenador/Staff es un actor independiente con acceso directo al módulo operativo, que comprende el registro de asistencia y la visualización del tablero. El Socio interactúa únicamente con el registro de asistencia.

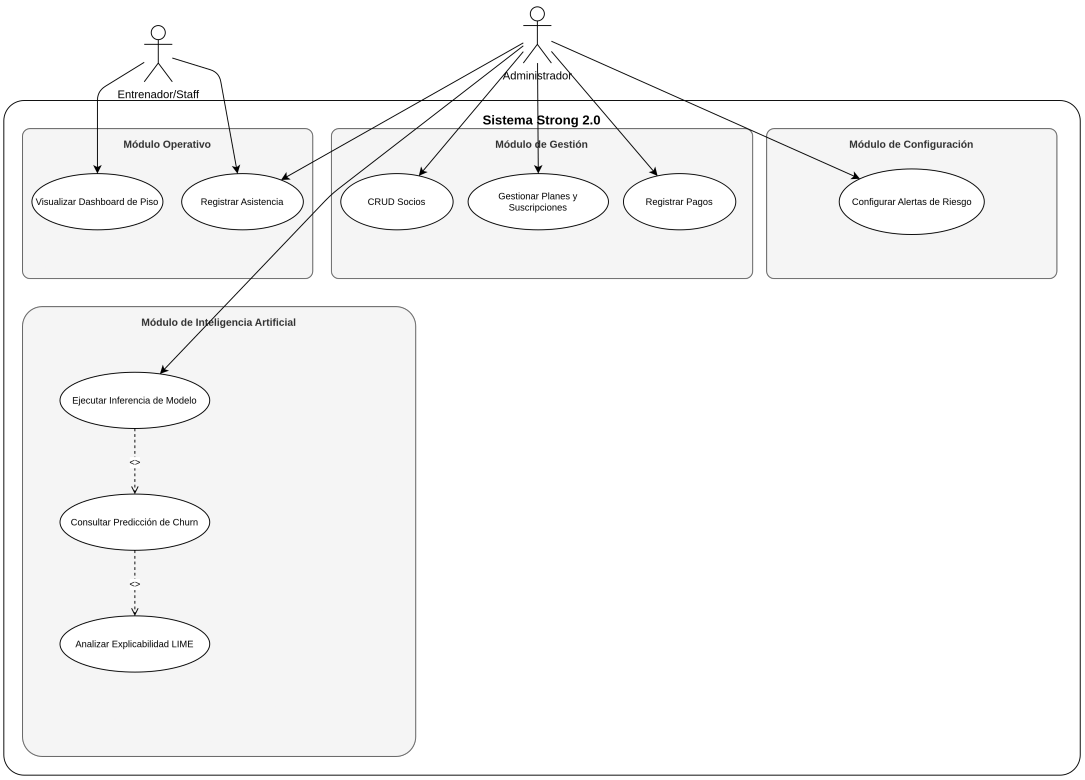


Figura 24: Diagrama de casos de uso del sistema Strong 2.0.

El sistema cuenta con seis casos de uso distribuidos entre los tres actores identificados. Las Tablas 40, 41 y 42 detallan la especificación de los tres casos de uso críticos seleccionados; la especificación completa de los casos de uso complementarios se presenta en el Anexo F.

Tabla 40: Especificación CU-01: Registrar Asistencia.

Actor	Administrador / Entrenador–Staff
Precondición	El usuario ha iniciado sesión con permisos de administrador o staff.
Flujo principal	1. El administrador accede al módulo de registro de asistencia. 2. El sistema muestra el formulario de ingreso de cédula. 3. El administrador ingresa el número de cédula del socio. 4. El sistema valida la cédula en la base de datos. 5. El sistema registra la asistencia con fecha y hora actuales. 6. El sistema confirma el ingreso con el nombre del socio.
Flujo alternativo	4a. Cédula no encontrada o membresía inactiva: el sistema muestra un mensaje de error. 5a. El socio ya registró ingreso ese día: el sistema notifica que el registro ya existe.
Postcondición	La asistencia queda registrada en la base de datos con fecha y hora.

Tabla 41: Especificación CU-02: Ejecutar Inferencia de Modelo.

Actor	Sistema (APScheduler) / Administrador
Precondición	Los archivos del modelo (`classifier.joblib`, `scaler.joblib`) están disponibles en el servidor.
Flujo principal	1. El planificador automático (APScheduler) activa el recálculo según el intervalo configurado en `frecuencia_inferencia_horas`. 2. El sistema obtiene la lista de socios activos. 3. Por cada socio, calcula: lifetime, frecuencia de asistencia, duración de membresía, edad y meses al vencimiento del contrato. 4. El sistema escala los datos con StandardScaler. 5. Gradient Boosting predice la probabilidad de deserción. 6. El sistema clasifica al socio en Alto, Medio o Bajo riesgo según umbrales configurables. 7. El sistema incluye CU-03 para generar la explicación individual. 8. El sistema persiste la predicción en la base de datos y registra la ejecución como automática.
Flujo alternativo	1a. El administrador activa el recálculo manualmente desde el dashboard; el sistema registra la ejecución indicando el usuario que la inició. 5a. Archivos `.joblib` no disponibles: el sistema retorna error HTTP 503 y registra la ejecución fallida en la bitácora.
Postcondición	Las predicciones de todos los socios activos están actualizadas en la base de datos.

Tabla 42: Especificación CU-03: Analizar Explicabilidad LIME.

Actor	Administrador (extiende CU-02)
Precondición	Existe una predicción calculada para el socio.
Flujo principal	1. El sistema toma los datos del socio en escala real, sin normalizar. 2. LIME genera perturbaciones alrededor del punto de datos del socio. 3. El módulo entrena un modelo lineal local sobre esas perturbaciones. 4. El sistema identifica los factores con mayor peso en la predicción. 5. Devuelve un diccionario ordenado por influencia con el factor y su peso numérico. 6. El administrador visualiza los factores en el dashboard.
Flujo alternativo	4a. Datos de asistencia insuficientes: el factor aparece con peso cero en la explicación.
Postcondición	La explicación LIME queda almacenada junto a la predicción del socio.

Modelado del proceso propuesto

Los cuatro procesos identificados en la situación actual se rediseñaron para su versión propuesta con Strong 2.0, incorporando automatización en cada etapa del flujo operativo.

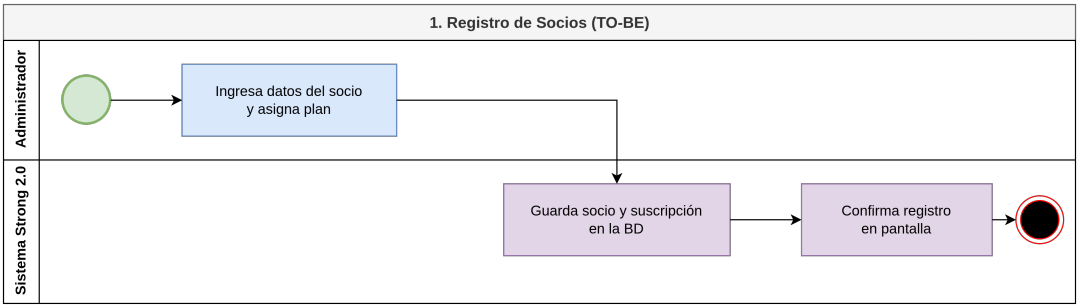


Figura 25: BPMN: Proceso de registro de socios (proceso propuesto)

El registro ya no depende de un cuaderno. El administrador ingresa los datos del socio directamente en el sistema y asigna el plan; Strong 2.0 persiste la información y

confirma el registro en pantalla (Figura 25).

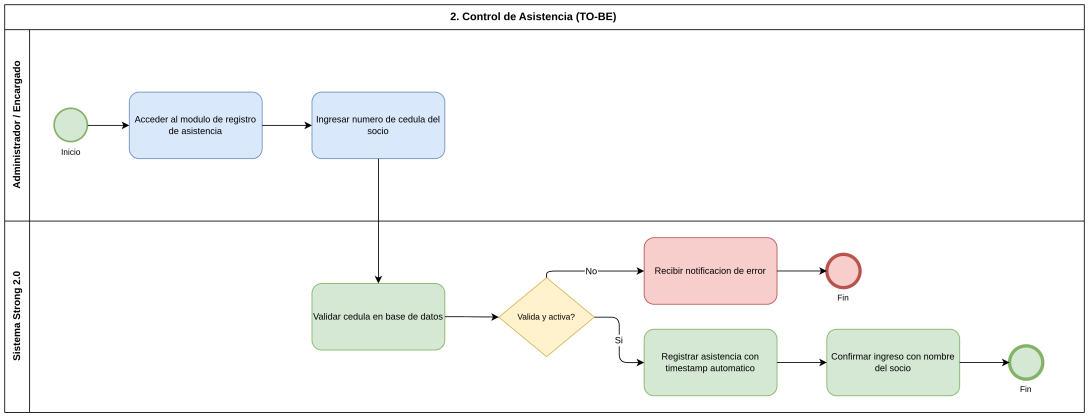


Figura 26: BPMN: Proceso de control de asistencia (proceso propuesto)

La asistencia es registrada por el administrador o encargado del gimnasio. El personal accede al módulo de registro de asistencia en Strong 2.0, ingresa el número de cédula del socio y el sistema valida su estado y registra la entrada con timestamp automático. Si la cédula no corresponde a un socio activo, se notifica el error en pantalla (Figura 26).

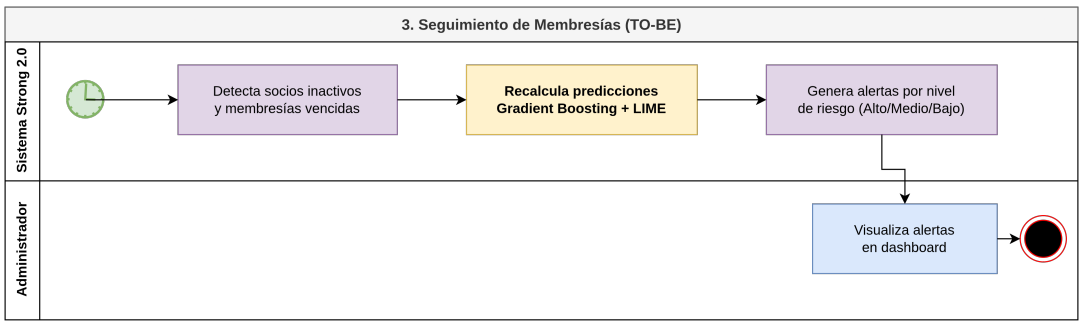


Figura 27: BPMN: Proceso de seguimiento de membresías (proceso propuesto)

El seguimiento deja de ser manual. El sistema detecta socios inactivos y membresías vencidas, recalcula las predicciones de forma automática y programada con el modelo de inteligencia artificial y genera alertas clasificadas por nivel de riesgo. El administrador las visualiza directamente en el dashboard sin necesidad de revisar ningún Excel ni ejecutar ninguna acción (Figura 27).

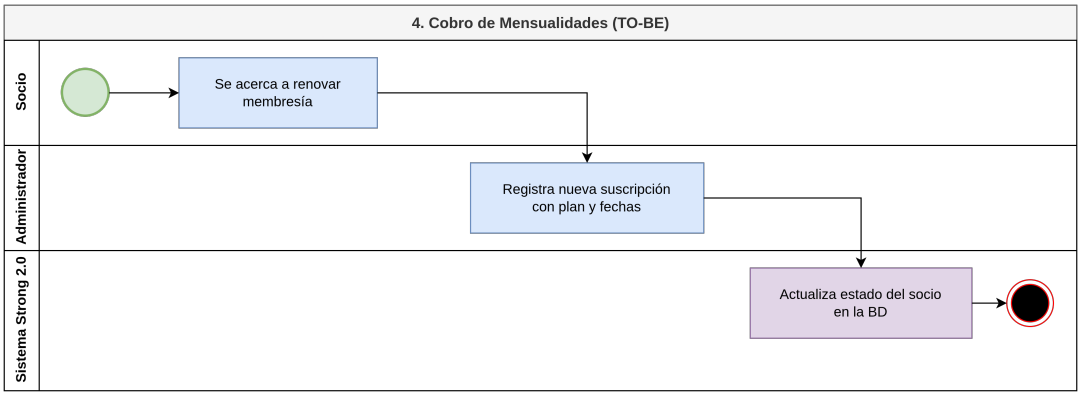


Figura 28: BPMN: Proceso de cobro de mensualidades (proceso propuesto)

El cobro tiene ahora un único canal. El administrador registra la renovación en el sistema con el plan y las fechas correspondientes; Strong 2.0 actualiza el estado del socio y deja trazabilidad del pago (Figura 28).

Selección de la metodología y tecnologías

La metodología de desarrollo y las tecnologías se seleccionaron mediante el modelo de decisión cuantitativo de Herbert Simon. En todas las matrices, las calificaciones siguen una escala de 1 a 5: 5 indica cumplimiento pleno del criterio y 1 indica ausencia de soporte nativo. Las calificaciones se asignaron con base en la documentación oficial de cada herramienta y los antecedentes identificados en la revisión de literatura; los pesos reflejan la prioridad relativa de cada criterio según los requisitos del sistema.

Elección de la metodologia de desarrollo: Se evaluaron tres metodologías considerando que el proyecto fue ejecutado por un único desarrollador, lo que hace prioritaria la viabilidad individual junto con la capacidad de adaptación ante los cambios típicos de los modelos de inteligencia artificial.

Tabla 43: Matriz de selección de la metodología de desarrollo.

Criterios	**Peso**	**SDD**	**Kanban**	**Solo Scrum**
Flexibilidad ante cambios	0.35	3	5	4
Entrega incremental de valor	0.35	4	3	5
Viabilidad para desarrollo individual	0.30	4	5	5
Puntaje Total	**1.00**	**3.65**	**4.30**	**4.65**

Como muestra la Tabla 43, se eligió Solo Scrum, una adaptación de Scrum para desarrollo individual que prescinde de las ceremonias de equipo manteniendo el ciclo de sprints como unidad de planificación por módulo. Este enfoque permitió organizar el avance del sistema de forma incremental y controlada a lo largo del cronograma establecido.

Elección del framework de desarrollo backend: Se evaluaron tres *frameworks* del ecosistema Python, ya que el modelo predictivo está desarrollado en ese lenguaje. Los criterios ponderan las necesidades reales del sistema: velocidad de entrega, solidez de seguridad y funcionalidades integradas que eviten construir desde cero el ORM, el panel de administración y el sistema de autenticación. Los antecedentes comparativos de cada opción se detallan en la Tabla 10.

Tabla 44: Matriz de selección del framework de desarrollo backend.

Criterios	**Peso**	**Django**	**FastAPI**	**Flask**
Velocidad de desarrollo	0.30	4	5	3
Seguridad y escalabilidad	0.30	5	4	3
Funcionalidades integradas (ORM/Admin)	0.40	5	3	2
Puntaje Total	**1.00**	**4.70**	**3.90**	**2.60**

La elección fue Django (Tabla 44). En velocidad de desarrollo, FastAPI obtiene la calificación más alta (5) por su mínima configuración inicial para exponer endpoints REST; Django obtiene 4 porque, si bien requiere mayor estructura de proyecto, sus herramientas integradas aceleran el desarrollo global; Flask obtiene 3 por exigir configu-

ración manual de cada componente desde el inicio. En seguridad y escalabilidad, Django obtiene 5 por sus mecanismos incorporados de protección CSRF, gestión de sesiones y el módulo de autenticación de Django REST Framework con soporte JWT; FastAPI obtiene 4 por ofrecer validación automática de datos y soporte HTTPS, pero depende de librerías externas para autenticación y autorización; Flask obtiene 3 al carecer de protecciones de seguridad nativas y requerir extensiones de terceros para cualquier control de acceso. En funcionalidades integradas, Django obtiene 5 por incluir ORM, panel de administración y sistema de autenticación listos para producción; FastAPI obtiene 3 al limitarse a la capa de API sin ORM ni panel administrativo nativos; Flask obtiene 2 por no incorporar ninguno de estos componentes, demandando integración manual de cada módulo.

Elección del framework de desarrollo frontend: Se evaluaron tres *frameworks* de desarrollo *frontend* considerando la necesidad de construir una interfaz administrativa reactiva que consuma la API REST de Django y presente los resultados del modelo predictivo de forma clara y eficiente. Los antecedentes comparativos de cada opción se detallan en la Tabla 11.

Tabla 45: Matriz de selección del framework de desarrollo frontend.

Criterios	**Peso**	**Next.js**	**Vue 3**	**Angular**
Velocidad de desarrollo	0.35	4	4	3
Integración con APIs REST	0.35	5	4	4
Mantenimiento y comunidad	0.30	5	4	4
Puntaje Total	**1.00**	**4.65**	**4.00**	**3.65**

Next.js obtuvo el mayor puntaje total (Tabla 45). En velocidad de desarrollo, Next.js y Vue 3 obtienen igual calificación (4): Next.js presenta mayor curva de aprendizaje inicial por los conceptos de renderizado en servidor y componentes de servidor, mientras que Vue 3 requiere configuración adicional para la integración con TypeScript y la gestión de estado; Angular obtiene 3 por su arquitectura más rígida y la curva de aprendizaje más pronunciada que imponen sus módulos, decoradores e inyección de dependencias. En integración con APIs REST, Next.js obtiene 5 por sus rutas de API

integradas y compatibilidad nativa con la API Fetch, que simplifican la comunicación con el backend de Django; Vue 3 y Angular obtienen 4 porque admiten axios y fetch sin restricciones, aunque no ofrecen un mecanismo equivalente a las rutas de API nativas de Next.js. En mantenimiento y comunidad, Next.js obtiene 5 por el respaldo del ecosistema React, la compatibilidad nativa con TypeScript y el soporte activo de Vercel; Vue 3 y Angular obtienen 4 por contar con comunidades maduras y soporte empresarial activo, aunque con ecosistemas de menor tamaño.

Elección de la biblioteca de optimización de hiperparámetros: La selección de Optuna como biblioteca de optimización de hiperparámetros se documenta en la Tabla 38, presentada en la sección de selección del modelo predictivo junto a la justificación técnica de su elección.

El desarrollo se organizó en 8 sprints de 2 semanas cada uno, abarcando el período de febrero a mayo. La semana 4 de mayo fue destinada a la revisión final y consolidación del documento de tesis, aunque la redacción fue un proceso continuo a lo largo de todo el proyecto. La Tabla 46 resume los períodos y entregables de cada sprint.

Tabla 46: Planificación de sprints del proyecto.

Sprint	Período	Entregable
1	Feb sem 1–2	Diagnóstico del gimnasio documentado
2	Feb sem 3–4	Requisitos funcionales y no funcionales
3	Mar sem 1–2	Dataset preparado y modelo seleccionado
4	Mar sem 3–4	Diseño del sistema
5	Abr sem 1–2	Gestión de socios funcional
6	Abr sem 3–4	Sistema web completo
7	May sem 1–2	IA integrada al sistema
8	May sem 2–3	Sistema validado

Cada sprint produjo entregables concretos documentados a continuación.

Sprint 1 – Diagnóstico: Se realizó el levantamiento de información del gimnasio, identificando los procesos manuales de registro de socios, control de asistencia y gestión de pagos. El diagnóstico consolidado, que incluye el análisis FODA de la

gestión administrativa, se presenta en la Tabla 33.

Sprint 2 – Requisitos: Se definieron los requisitos funcionales y no funcionales del sistema a partir del diagnóstico previo, e inició la investigación de algoritmos de predicción de deserción aplicables al contexto del gimnasio. El conjunto completo de requisitos se detalla en las Tablas 34 y 35.

Sprint 3 – Modelo y datos: Se seleccionó el modelo predictivo más adecuado mediante análisis comparativo de algoritmos y se ejecutó el preprocesamiento del dataset, incluyendo limpieza, normalización e ingeniería de variables. El resultado de este sprint se refleja en el análisis comparativo de la Tabla 48 y la importancia de variables de la Figura 21.

Sprint 4 – Diseño: Se definió la arquitectura de tres capas del sistema, el modelo relacional de la base de datos y los prototipos de interfaz, estableciendo la estructura técnica sobre la cual se desarrollaría la plataforma.

El sistema adopta una arquitectura de tres capas que separa la interfaz, la lógica de negocio y la persistencia (Figura 29). Esta separación permite actualizar el motor de IA de forma independiente sin interrumpir la operación del gimnasio.

La capa de presentación, implementada en Next.js 15, expone el panel administrativo, que consume la API REST mediante un cliente HTTP con interceptores JWT para todas las operaciones del sistema, incluyendo el registro de asistencia.

La capa de lógica, construida con Django 5.2 y Django REST Framework, centraliza toda la lógica de negocio detrás del módulo de autenticación JWT. Todas las operaciones, incluido el registro de asistencia, requieren autenticación; solo el personal autorizado puede acceder al sistema. El Planificador APScheduler gestiona dos jobs autónomos: el primero ejecuta el recálculo masivo de predicciones según el intervalo definido por el administrador en la configuración del sistema; el segundo verifica diariamente las condiciones de reentrenamiento del modelo y, cuando se cumplen, lanza el pipeline de reentrenamiento. El Pipeline ML carga los modelos entrenados desde archivos `.joblib` en el servidor, ejecuta Gradient Boosting para calcular la probabilidad de

deserción de cada socio y genera la explicación individual mediante LIME.

La capa de persistencia, gestionada con PostgreSQL, almacena los datos operativos en tablas relacionales normalizadas. Los resultados del modelo de IA se guardan en formato `JSONB`, lo que preserva el historial de predicciones y las explicaciones XAI por socio.

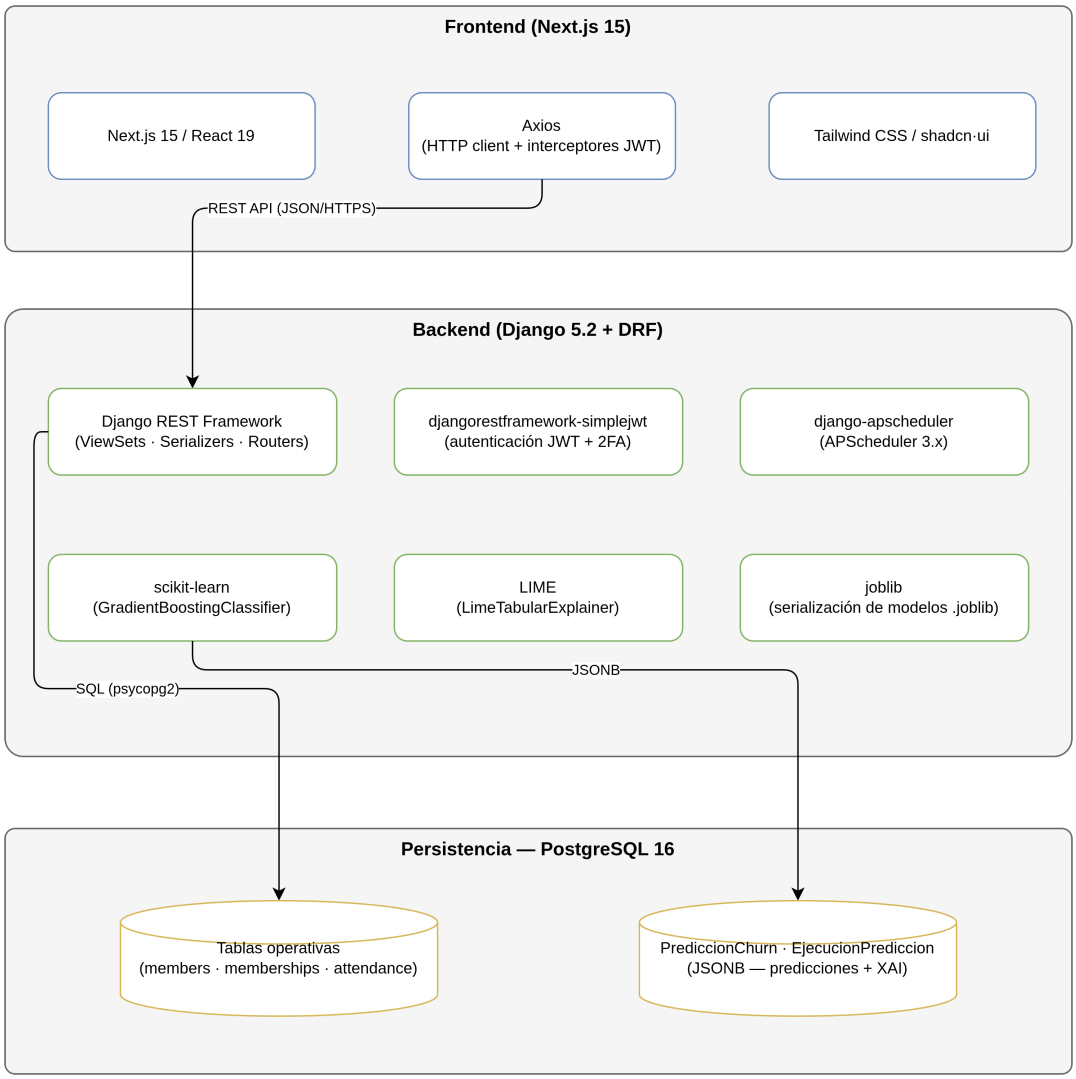


Figura 29: Arquitectura de tres capas del sistema Strong 2.0.

En la Figura 30 se detalla cómo se organizan los datos en PostgreSQL. El esquema garantiza la integridad operativa y la trazabilidad de las inferencias del modelo. La tabla `PrediccionChurn` almacena la predicción vigente de cada socio con restricción `UNIQUE` sobre `socio_id`, de modo que cada ejecución del pipeline sobreescribe el resultado anterior; el campo `explicacion_xai` en formato `JSONB` persiste la

explicación individual generada por LIME. La tabla `EjecucionPrediccion` actúa como bitácora de cada corrida del pipeline, registrando fecha, estado y cantidad de socios procesados, lo que permite auditar el historial de ejecuciones con independencia de los resultados vigentes.

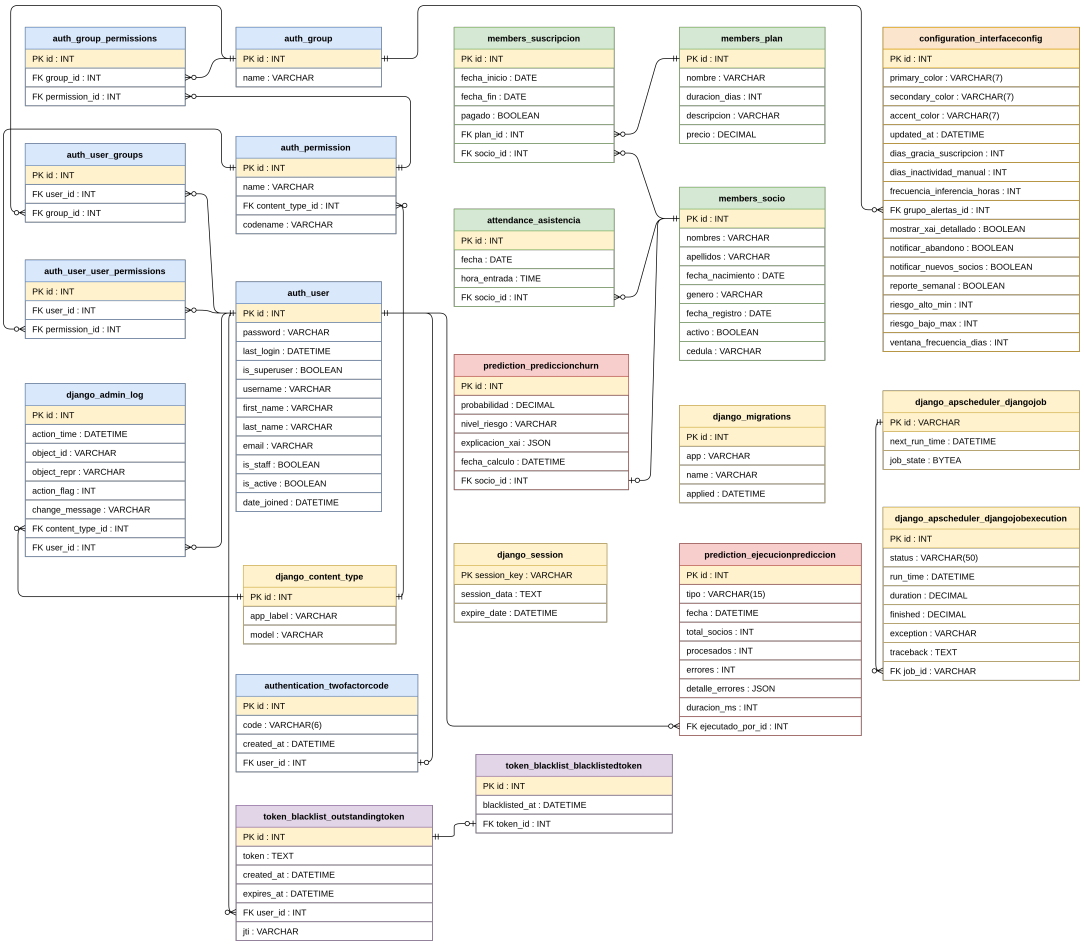


Figura 30: Modelo lógico de datos para la gestión y analítica.

La Figura 31 detalla el flujo de información desde el registro de asistencia realizado por el administrador hasta la generación de alertas de riesgo. El registro de asistencia alimenta la base de datos operativa, cuyos datos procesa el Pipeline ML para calcular la probabilidad de deserción. El resultado, junto con la explicación generada por LIME, se almacena en la base de datos y queda disponible para el administrador en el panel de control.

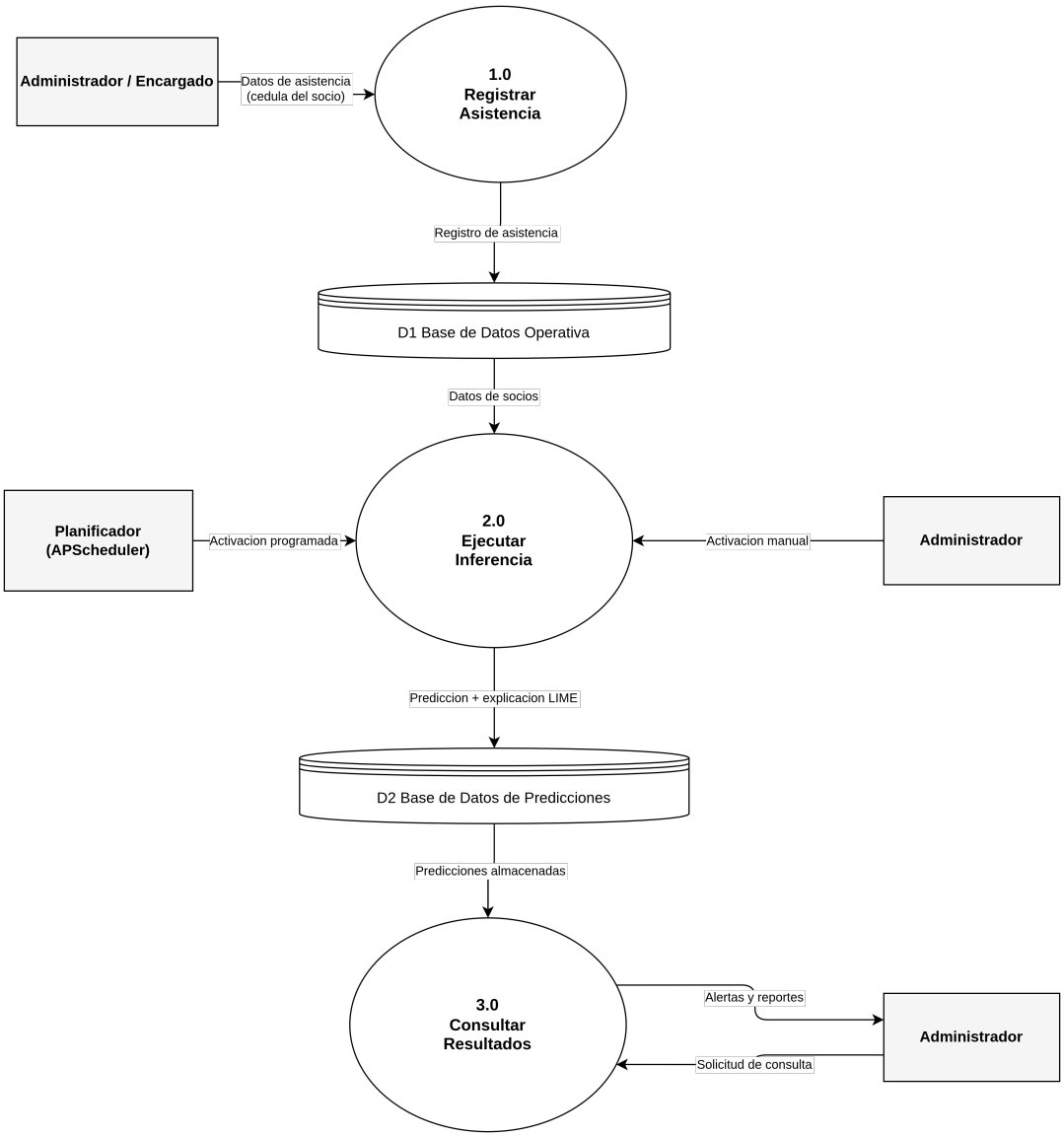


Figura 31: Flujo de información y procesamiento de predicciones.

Sprint 5 – Backend usuarios y suscripciones: Se implementó el módulo de gestión de socios y suscripciones en el backend, incluyendo el sistema de autenticación, registro de membresías y alertas automáticas de vencimiento (Figura 32).

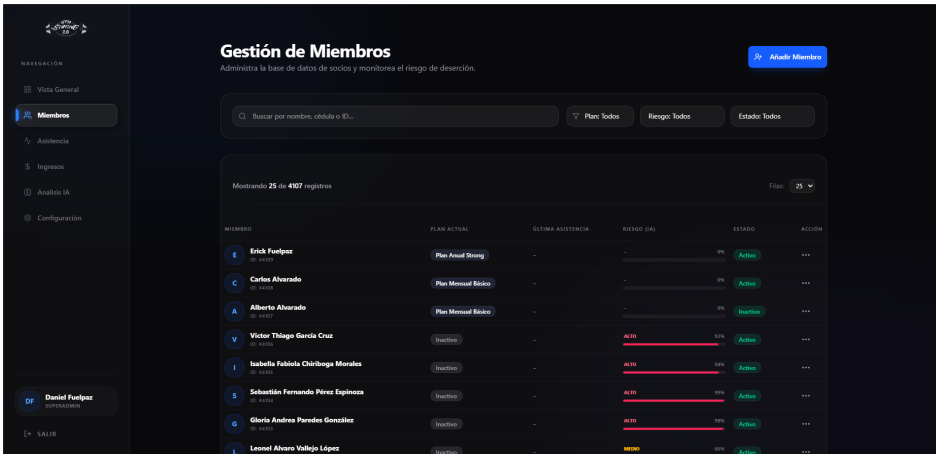


Figura 32: Sprint 5: Módulo de gestión de socios y suscripciones del sistema. Los registros mostrados corresponden al conjunto de datos de demostración, integrado por 150 fichas del gimnasio Strong 2.0 y 4.000 registros del *dataset* público de Kaggle [1], utilizado para ampliar el volumen de entrenamiento del modelo predictivo.

Sprint 6 – Sistema web completo: Se desarrolló el módulo de control de asistencia (Figura 33), el módulo de gestión de pagos (Figuras 34 y 35) y el panel administrativo frontend, completando el sistema web con todas las funcionalidades operativas integradas.

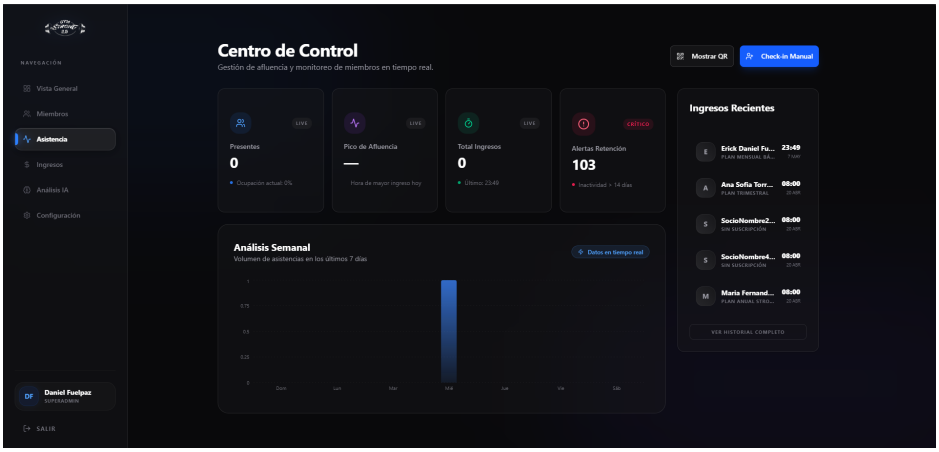


Figura 33: Sprint 6: Centro de Control del módulo de asistencia del sistema web.

El Centro de Control es la vista principal del módulo de asistencia y presenta en tiempo real cuatro indicadores clave: miembros presentes, hora de mayor ingreso del día, total de ingresos del día y alertas críticas de retención. El indicador de hora de mayor ingreso corresponde a la franja horaria con mayor volumen de entradas registradas en la jornada en curso, calculado dinámicamente a partir de los registros del día actual;

su objetivo es identificar el pico de afluencia para apoyar la gestión operativa del establecimiento. Un gráfico de barras semanal muestra la evolución del volumen de asistencia en los últimos siete días. El panel lateral registra los ingresos recientes con nombre del socio, plan activo y hora de entrada. El módulo ofrece *check-in* mediante código QR o de forma manual desde el mismo panel.

Figura 34: Sprint 6: Módulo de gestión de pagos: registro de cobros.

El módulo de Gestión de Ingresos, correspondiente al requerimiento funcional RF-02 (Control de membresías), consolida los principales indicadores financieros: ingresos totales, MRR, ARPU y tasa de abandono. Un gráfico de crecimiento mensual desglosa los ingresos por tipo de plan durante los últimos seis meses. La tabla de Control de Facturación lista los registros con filtros por plan y estado de pago. Desde esta vista el administrador puede registrar nuevos cobros con el botón *Crear Pago*.

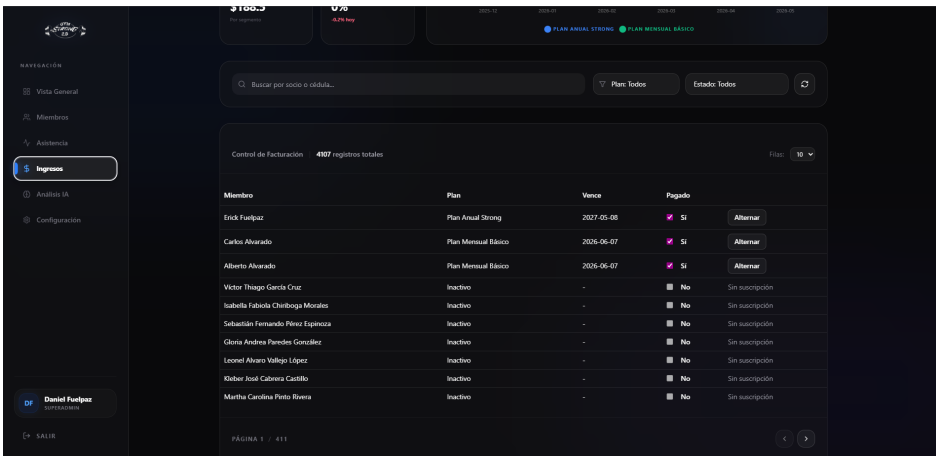


Figura 35: Sprint 6: Módulo de gestión de pagos: historial de transacciones.

La tabla de facturación muestra por fila el miembro, el plan contratado, la fecha de vencimiento y el estado de pago. Los registros con suscripción activa aparecen marcados como pagados; los socios sin plan figuran como inactivos. El administrador puede alternar el estado de pago directamente desde la tabla sin necesidad de editar el registro completo. La paginación gestiona los registros distribuidos en páginas con navegación secuencial.

Sprint 7 – Integración de la IA: Se entrenó el modelo Gradient Boosting con el dataset preparado y se integró al sistema web mediante una API REST interna. Este sprint conectó el modelo con la interfaz administrativa: queda alojado en el servidor Django, expone sus predicciones a través de un endpoint REST consumido por el frontend en Next.js y se ejecuta de forma automática mediante APScheduler. Además se implementó autenticación con JSON Web Tokens (JWT) para proteger el acceso al panel. La actividad de desarrollo de este sprint se resume en la Figura 36. El administrador visualiza en tiempo real el nivel de riesgo de cada socio y los factores explicativos de LIME.

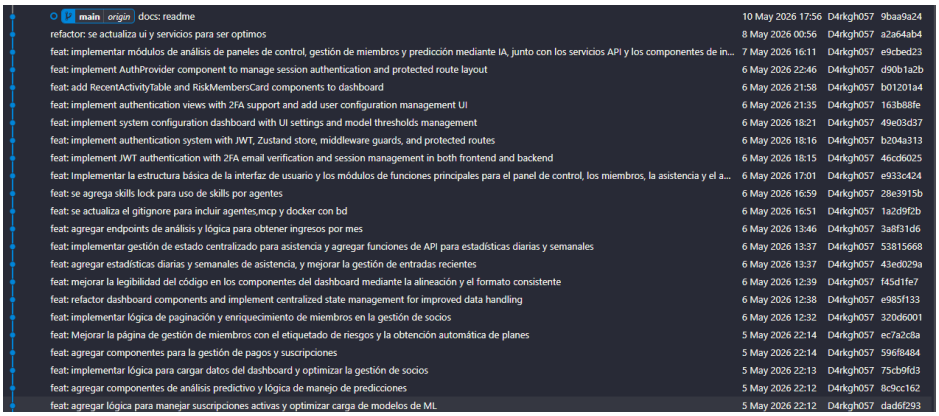


Figura 36: Sprint 7: Actividad de desarrollo: integración del modelo de IA y autenticación JWT.

La integración del modelo aporta un valor que la observación directa no puede replicar. Un administrador puede notar que un socio dejó de asistir, pero no puede evaluar simultáneamente seis variables interrelacionadas, como la antigüedad, la frecuencia histórica, la frecuencia reciente, la edad, los meses restantes de contrato y la duración del plan, para todos los socios activos al mismo tiempo. El modelo Gradient Boosting captura interacciones no lineales entre estas variables que no son perceptibles de forma

intuitiva: por ejemplo, un socio con alta frecuencia histórica pero baja frecuencia reciente y contrato próximo a vencer representa un riesgo elevado que ninguna regla simple detectaría con consistencia. Además, LIME genera una explicación individual por cada predicción, de modo que el administrador no recibe solo un porcentaje de riesgo, sino una guía de intervención concreta que identifica qué factor específico impulsó la alerta para ese socio en particular.

En este sprint se integró el motor predictivo con el módulo de registro de asistencia. El diagrama de secuencia (Figura 37) muestra cómo el software valida al socio y activa el análisis de riesgo automáticamente.

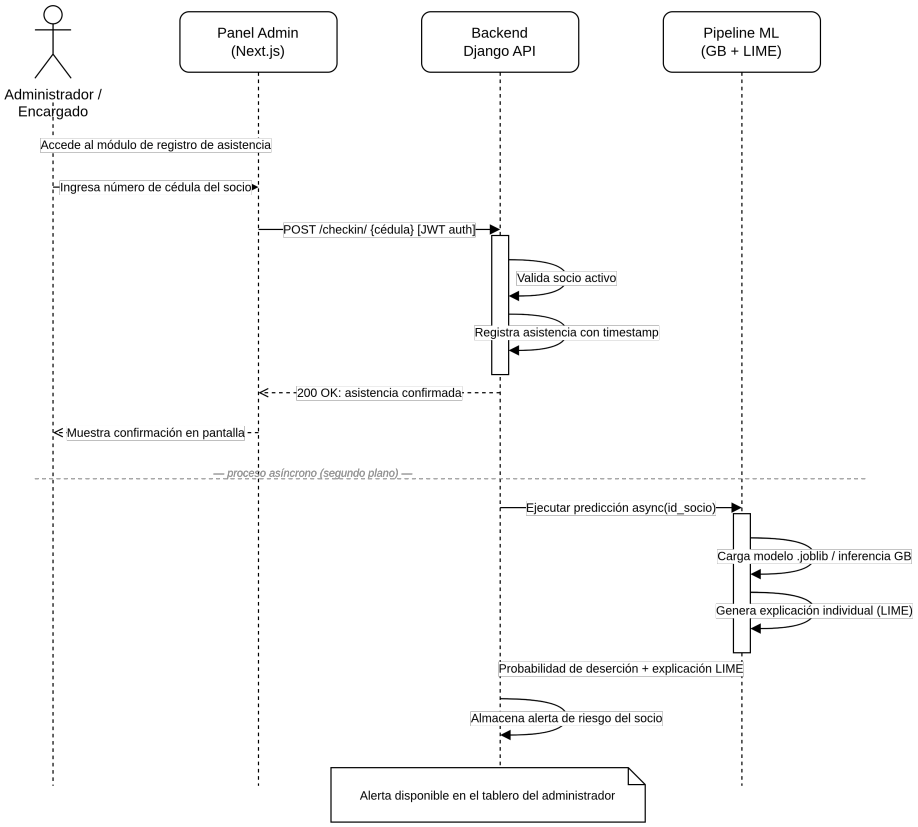


Figura 37: Diagrama de secuencia: Registro y análisis predictivo automático.

El diagrama sintetiza la interacción automatizada entre el módulo de asistencia y el motor predictivo. Desde el momento en que el administrador registra el ingreso de un socio, el sistema desencadena una cadena de procesos en segundo plano que culmina en la generación de una alerta de riesgo personalizada. Los dos procesos centrales de este flujo son:

Procesamiento de datos:

El sistema ajusta la información de asistencia para que el modelo la entienda y lanza la predicción.

Uso de XAI:

Mediante LIME, el administrador recibe un informe sobre qué factores pesan más en el riesgo de deserción del socio.

Como muestra la Figura 38, el proceso no requiere intervención manual. El administrador siempre tiene los datos listos en su tablero de control.

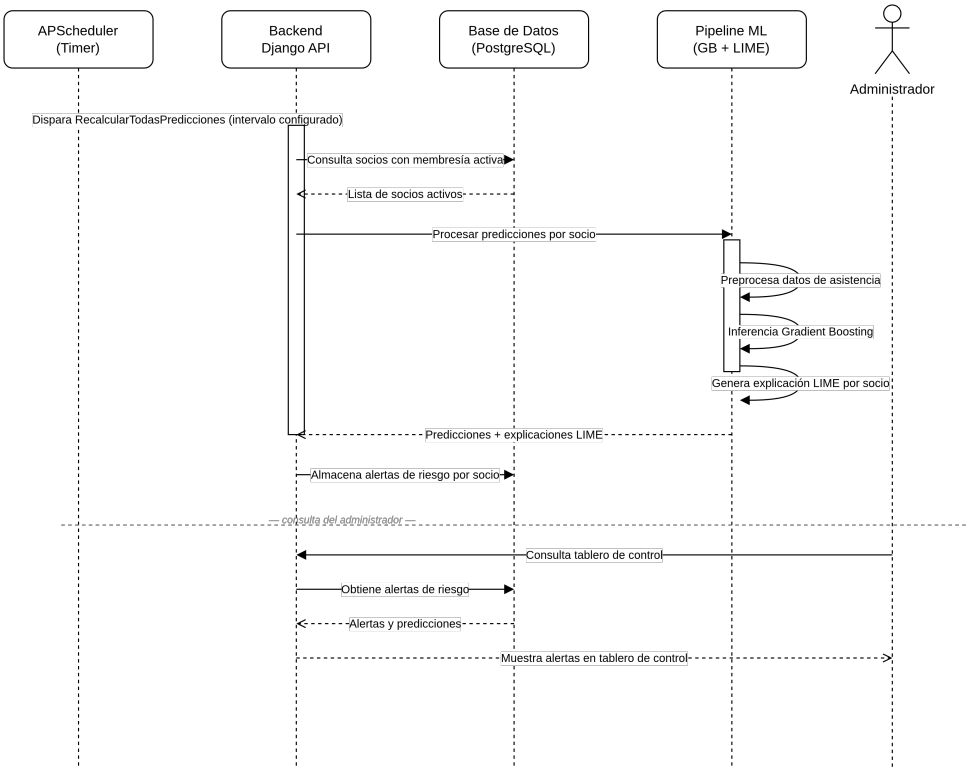


Figura 38: Interacción del sistema para generar alertas de riesgo.

Para garantizar que las predicciones reflejen siempre el estado más reciente de los socios, se implementó un planificador automático mediante *django-apscheduler* (APScheduler 3.x). El administrador configura el intervalo de recálculo directamente en el panel de configuración del sistema (`frecuencia_inferencia_horas`); Strong 2.0 ejecuta el proceso *RecalcularTodasPredicciones* de forma autónoma según ese intervalo, persiste los trabajos programados en la base de datos y los reanuda

automáticamente tras un reinicio del servidor. El registro de cada ejecución indica si fue iniciada por el planificador o manualmente por el administrador, lo que permite auditar el historial de inferencias sin ambigüedad.

Los resultados de cada ejecución se persisten en dos tablas. `PrediccionChurn` almacena una predicción vigente por socio: la restricción `UNIQUE (socio_id)` garantiza que cada ejecución sobreescriba el registro anterior, de modo que el dashboard siempre refleja el estado más reciente sin acumular histórico por socio. Cada registro contiene la probabilidad de deserción calculada por el modelo, el nivel de riesgo visual asignado según los umbrales configurados y el campo `explicacion_xai`, un objeto JSON generado por LIME que detalla qué variables y en qué dirección influyeron en esa predicción individual. `EjecucionPrediccion` actúa como bitácora de auditoría: registra cada corrida del proceso con el tipo de disparo (planificador automático o ejecución manual), la fecha, el total de socios procesados, los errores ocurridos con su detalle y la duración en milisegundos, lo que permite al administrador verificar que el motor predictivo opera con normalidad sin necesidad de revisar logs del servidor.

Con el fin de mantener la vigencia del modelo a medida que se acumulan nuevos datos, se implementó un segundo job en el mismo planificador: el proceso *Reentrenar-Modelo*. Este job se verifica diariamente, pero el reentrenamiento efectivo se activa únicamente cuando se cumplen dos condiciones de forma simultánea: haber transcurrido al menos 180 días desde el último reentrenamiento desplegado y disponer de al menos 150 registros etiquetados acumulados desde ese momento. Una vez activado el pipeline, el sistema reconstruye el conjunto de datos desde la base de datos, divide los registros en entrenamiento y validación con proporción 80/20, ajusta un nuevo escalador y entrena un clasificador Gradient Boosting con los mismos hiperparámetros del modelo original. El candidato se despliega únicamente si supera al modelo en producción en la métrica ROC AUC calculada sobre el conjunto de validación; de lo contrario, los artefactos originales permanecen sin modificación. Cada intento queda registrado en `ModelRetrainingLog` con la fecha, el número de registros disponibles, los valores de ROC AUC del modelo anterior y del candidato, y el motivo del resultado, lo que permite auditar el historial de reentrenamientos con precisión.

Las Figuras 39 y 40 muestran las dos secciones principales del módulo de configuración: la configuración de alertas del sistema y el catálogo de membresías disponibles en el gimnasio. Los umbrales de probabilidad (Riesgo Bajo y Riesgo Crítico) son parámetros de visualización: determinan cómo se etiqueta la probabilidad de deserción que devuelve el modelo en los niveles de alerta del dashboard, sin modificar el proceso de predicción ni el threshold interno del clasificador. Esta configuración da cumplimiento al RF-05, que establece que el sistema debe emitir avisos automáticos según los límites de probabilidad definidos por el administrador.

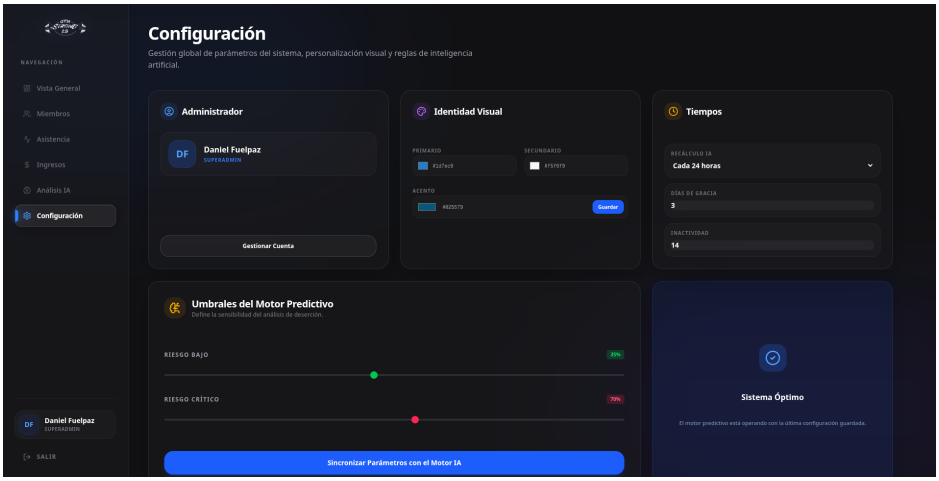


Figura 39: Panel de configuración: umbrales de alerta (Riesgo Bajo y Riesgo Crítico), frecuencia de recálculo IA, días de gracia e inactividad manual.

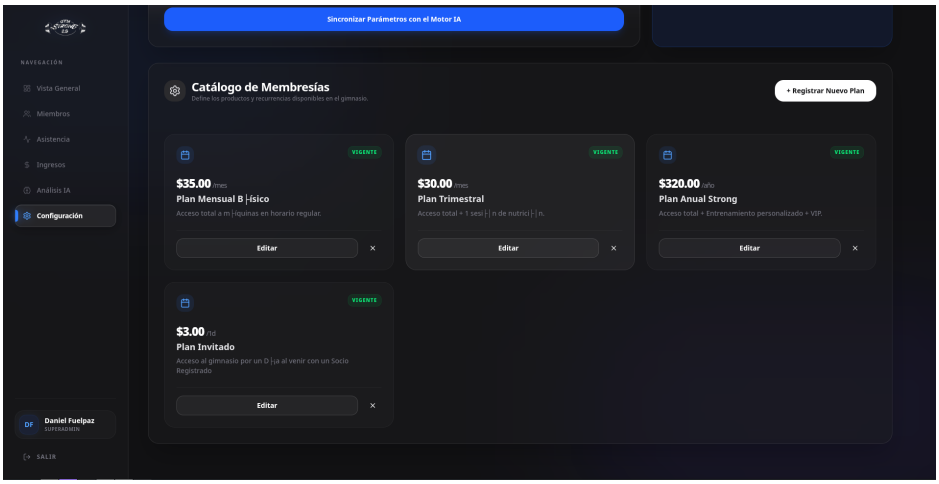


Figura 40: Panel de configuración: catálogo de membresías disponibles en el gimnasio.

El panel incluye también el parámetro `dias_inactividad_manual`, que define un umbral de alerta basado en conteo simple de días sin asistencia. Este mecanismo

opera de forma independiente al modelo: no modifica las variables de entrenamiento ni la definición de la clase objetivo, sino que genera una alerta complementaria cuando un socio supera el umbral configurado sin que el modelo haya emitido previamente una predicción de riesgo.

Sprint 8 – Pruebas: Se ejecutaron pruebas de integración entre el motor de IA y la interfaz web, validando el cumplimiento de los requisitos funcionales y no funcionales definidos.

Para verificar que la plataforma Strong 2.0 funcione de manera correcta y confiable, se ejecutó un plan de pruebas que abarcó pruebas unitarias, de integración, caja blanca y caja negra. Las pruebas validaron la ausencia de errores técnicos y aseguraron la integridad del flujo de datos entre el motor de Inteligencia Artificial y la interfaz administrativa.

La Tabla 47 detalla las pruebas ejecutadas, la técnica aplicada y el resultado obtenido:

Tabla 47: Matriz de validación técnica y control de calidad.

Tipo de Prueba	Herramienta	Escenario de Validación	Estado
Unitarias	Pytest / Vitest / Cov.	Validación del motor de predicción (66 % de cobertura global) y lógica de traducción XAI.	**Éxito**
Integración	Bruno / Django Test	Validación del flujo de datos desde el registro de entrada hasta la persistencia en PostgreSQL en formato JSONB.	**Éxito**
Caja Blanca	Coverage.py	Análisis de cobertura de rutas críticas en el middleware de seguridad y procesamiento de inferencia.	**Éxito**
Caja Negra	Testing Manual / Vitest	Simulación de usuarios y validación de generación de *hallazgos* automáticos (ImpactFactors).	**Éxito**

Las 51 pruebas automatizadas se ejecutaron satisfactoriamente y confirmaron que el sistema cumple los estándares de calidad requeridos para su despliegue. 35 casos correspondieron al backend, ejecutados con *Pytest* sobre el núcleo de predicción, los

endpoints de la API REST y los middlewares de seguridad. 16 casos correspondieron al frontend, ejecutados con *Vitest* para validar la lógica de presentación de los resultados del modelo de IA. Los dos grupos cubrieron desde el comportamiento de cada componente de forma independiente hasta el flujo completo del sistema:

Pruebas unitarias: Se enfocaron en validar el funcionamiento de cada componente de forma independiente. En el backend, mediante *Pytest*, se alcanzó un 66 % de cobertura global del código (Figuras 41 y 42). La fiabilidad del núcleo de predicción quedó respaldada según los estándares industriales modernos [16]. En el frontend, *Vitest* superó sus 16 casos de prueba y aseguró la integridad de la lógica de negocio. El componente *xaiLabels* fue el más relevante: verificó la traducción de los rangos numéricos generados por LIME, como "$1{,}5 <$ feature $\leq 3{,}0$", a descriptores legibles para el administrador. También confirmó que el semáforo de riesgo se visualizara correctamente en *RiskMembersCard*, tal como se muestra en la Figura 43.

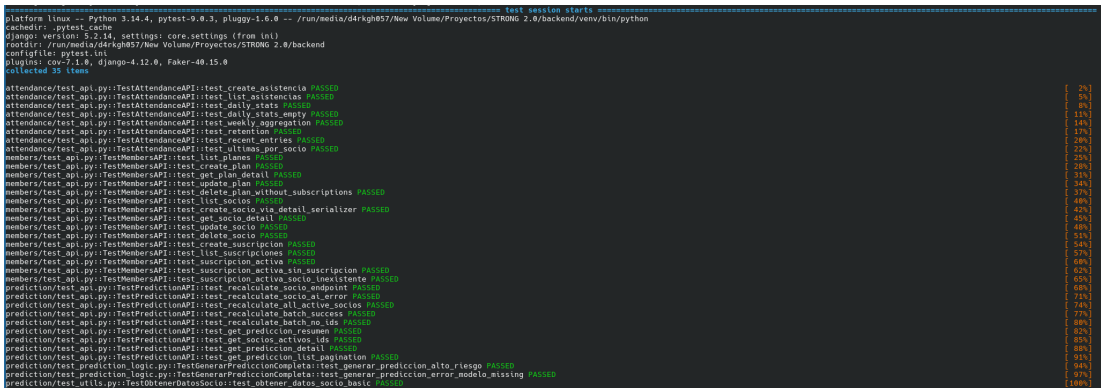
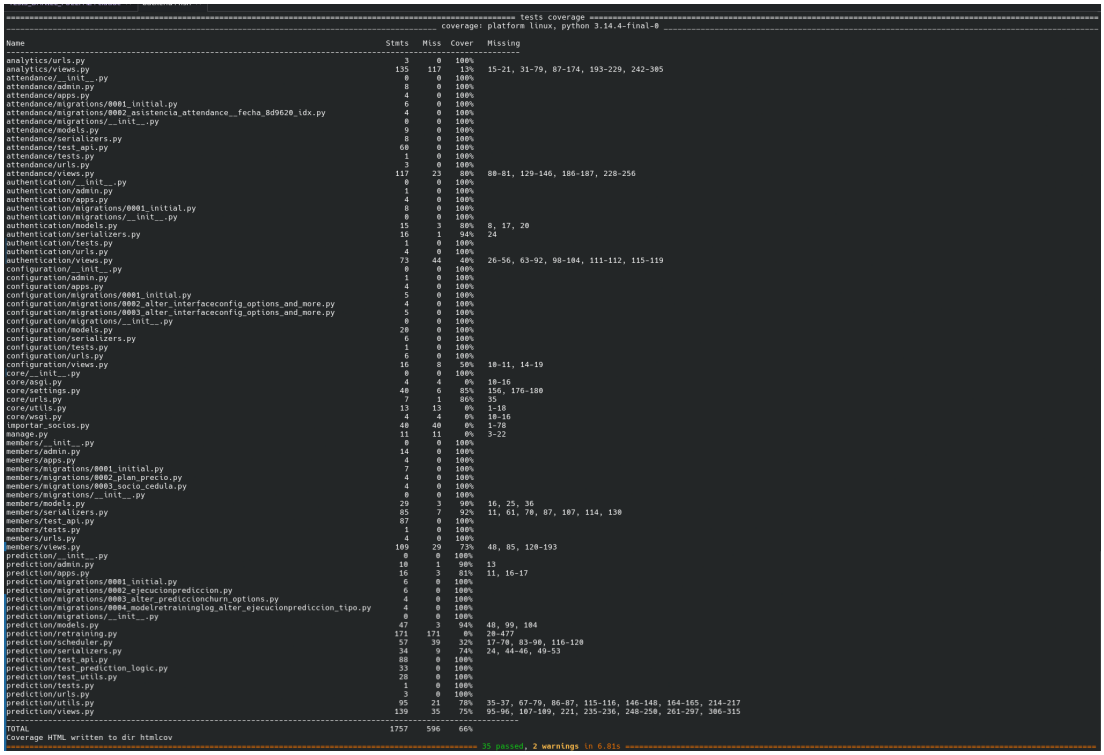


Figura 41: Resultados de pruebas unitarias en el backend: 35 casos aprobados.



```
============================================================ tests coverage ============================================================
_____ coverage: platform linux, python 3.14.4-final-0 _____

Name                                                                                                                  Stmts   Miss  Cover   Missing
---------------------------------------------------------------------------------------------------------------------------------------------------
analytics/urls.py                                                                                                         3      0   100%
analytics/views.py                                                                                                      135    117    13%   15-21, 31-79, 87-174, 193-229, 242-305
attendance/__init__.py                                                                                                    0      0   100%
attendance/admin.py                                                                                                       8      0   100%
attendance/apps.py                                                                                                        4      0   100%
attendance/migrations/0001_initial.py                                                                                     6      0   100%
attendance/migrations/0002_asistencia_attendance__fecha_8d9620_idx.py                                                     4      0   100%
attendance/migrations/__init__.py                                                                                         0      0   100%
attendance/models.py                                                                                                      9      0   100%
attendance/serializers.py                                                                                                 8      0   100%
attendance/test_api.py                                                                                                   60      0   100%
attendance/tests.py                                                                                                       1      0   100%
attendance/urls.py                                                                                                        3      0   100%
attendance/views.py                                                                                                     117     23    80%   80-81, 129-146, 186-187, 228-256
authentication/__init__.py                                                                                                0      0   100%
authentication/admin.py                                                                                                   1      0   100%
authentication/apps.py                                                                                                    4      0   100%
authentication/migrations/0001_initial.py                                                                                 8      0   100%
authentication/migrations/__init__.py                                                                                     0      0   100%
authentication/models.py                                                                                                 15      3    80%   8, 17, 20
authentication/serializers.py                                                                                            16      1    94%   24
authentication/tests.py                                                                                                   1      0   100%
authentication/urls.py                                                                                                    4      0   100%
authentication/views.py                                                                                                  73     44    40%   26-56, 63-92, 98-104, 111-112, 115-119
configuration/__init__.py                                                                                                 0      0   100%
configuration/admin.py                                                                                                    1      0   100%
configuration/apps.py                                                                                                     4      0   100%
configuration/migrations/0001_initial.py                                                                                  5      0   100%
configuration/migrations/0002_alter_interfaceconfig_options_and_more.py                                                   4      0   100%
configuration/migrations/0003_alter_interfaceconfig_options_and_more.py                                                   5      0   100%
configuration/migrations/__init__.py                                                                                      0      0   100%
configuration/models.py                                                                                                  20      0   100%
configuration/serializers.py                                                                                              6      0   100%
configuration/tests.py                                                                                                    1      0   100%
configuration/urls.py                                                                                                     6      0   100%
configuration/views.py                                                                                                   16      8    50%   10-11, 14-19
core/__init__.py                                                                                                          0      0   100%
core/asgi.py                                                                                                              4      4     0%   10-16
core/settings.py                                                                                                         40      6    85%   156, 176-180
core/urls.py                                                                                                              7      1    86%   35
core/utils.py                                                                                                            13     13     0%   1-18
core/wsgi.py                                                                                                              4      4     0%   10-16
importar_socios.py                                                                                                       40     40     0%   1-78
manage.py                                                                                                                11     11     0%   3-22
members/__init__.py                                                                                                       0      0   100%
members/admin.py                                                                                                         14      0   100%
members/apps.py                                                                                                           4      0   100%
members/migrations/0001_initial.py                                                                                        7      0   100%
members/migrations/0002_plan_precio.py                                                                                    4      0   100%
members/migrations/0003_socio_cedula.py                                                                                   4      0   100%
members/migrations/__init__.py                                                                                            0      0   100%
members/models.py                                                                                                        29      3    90%   16, 25, 36
members/serializers.py                                                                                                   85      7    92%   11, 61, 70, 87, 107, 114, 130
members/test_api.py                                                                                                      87      0   100%
members/tests.py                                                                                                          1      0   100%
members/urls.py                                                                                                           4      0   100%
members/views.py                                                                                                        109     29    73%   40, 85, 120-193
prediction/__init__.py                                                                                                    0      0   100%
prediction/admin.py                                                                                                      10      1    90%   13
prediction/apps.py                                                                                                       16      3    81%   11, 16-17
prediction/migrations/0001_initial.py                                                                                     6      0   100%
prediction/migrations/0002_ejecucionprediccion.py                                                                         6      0   100%
prediction/migrations/0003_alter_prediccionchurn_options.py                                                               4      0   100%
prediction/migrations/0004_modelretrainginglog_alter_ejecucionprediccion_tipo.py                                          4      0   100%
prediction/migrations/__init__.py                                                                                         0      0   100%
prediction/models.py                                                                                                     47      3    94%   40, 99, 104
prediction/retraining.py                                                                                                171    171     0%   20-477
prediction/scheduler.py                                                                                                  57     39    32%   17-78, 83-90, 116-120
prediction/serializers.py                                                                                                34      9    74%   24, 44-46, 49-53
prediction/test_api.py                                                                                                   88      0   100%
prediction/test_prediction_logic.py                                                                                      33      0   100%
prediction/test_utils.py                                                                                                 28      0   100%
prediction/tests.py                                                                                                       1      0   100%
prediction/urls.py                                                                                                        3      0   100%
prediction/utils.py                                                                                                      95     21    78%   35-37, 67-79, 86-87, 115-116, 146-148, 164-165, 214-217
prediction/views.py                                                                                                     139     35    75%   95-96, 107-109, 221, 235-236, 248-258, 261-297, 306-315
---------------------------------------------------------------------------------------------------------------------------------------------------
TOTAL                                                                                                                  1757    596    66%
Coverage HTML written to dir htmlcov
========================================================== 35 passed, 2 warnings in 6.81s ==========================================================
```

Figura 42: Resumen de cobertura de código tras la ejecución de pruebas unitarias.

```
> frontend@0.1.0 test
> vitest

The CJS build of Vite's Node API is deprecated. See https://vite.dev/guide/troubleshooting.html#vite-cjs-node-api-deprecated for more details.

 DEV  v2.1.9 E:/Proyectos/STRONG 2.0/frontend

 ✓ src/features/dashboard/components/KPICards.test.tsx (2)
 ✓ src/features/dashboard/components/RiskMembersCard.test.tsx (4)
 ✓ src/features/predicciones/components/ImpactFactors.test.tsx (5)
 ✓ src/features/predicciones/utils/xaiLabels.test.ts (5)

 Test Files  4 passed (4)
      Tests  16 passed (16)
   Start at  20:32:22
   Duration  3.56s (transform 189ms, setup 888ms, collect 1.71s, tests 152ms, environment 6.62s, prepare 2.89s)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Figura 43: Resultados de pruebas unitarias en el frontend: 16 casos aprobados mediante Vitest.

Pruebas de integración: Validaron la comunicación entre los servicios de Django, el motor predictivo y la persistencia en PostgreSQL. Se verificó que los objetos analíticos en formato *JSONB* mantengan su integridad al ser consultados desde el frontend, asegurando que no existan rupturas en el flujo de información entre capas. Las Figuras 44, 45, 46 y 47 muestran los pasos verificados.

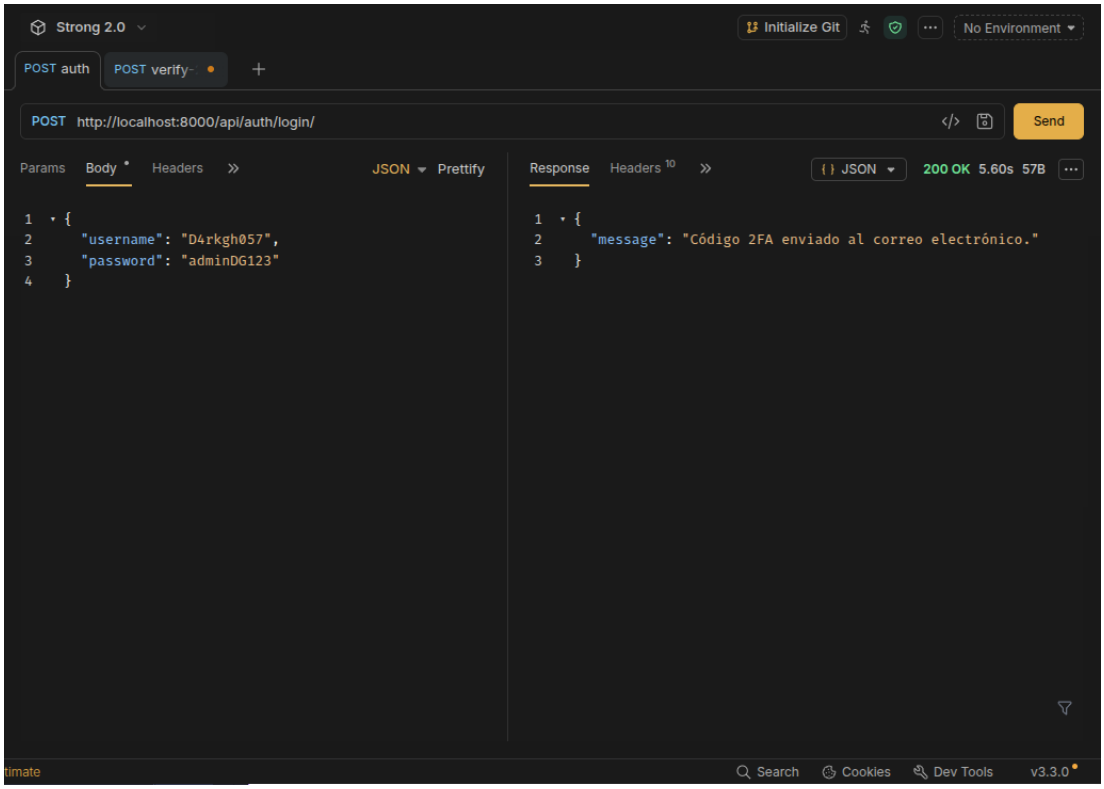


Figura 44: Prueba de integración: autenticación JWT, paso 1: login.

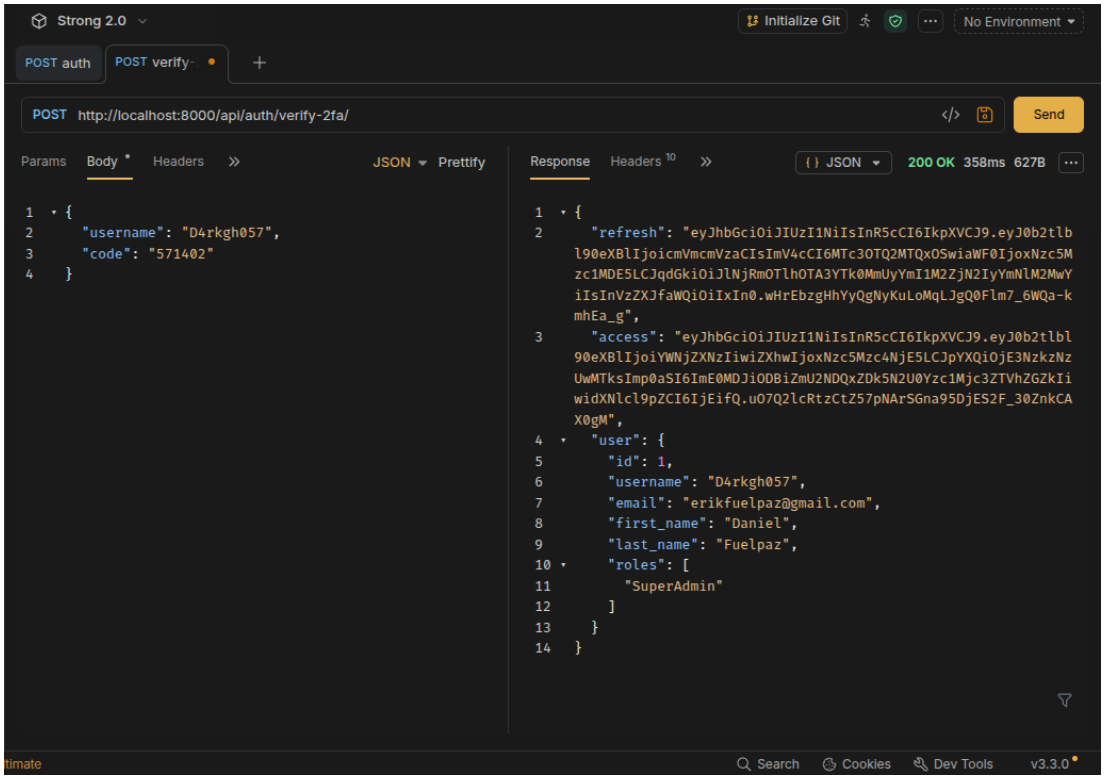


Figura 45: Prueba de integración: verificación 2FA y obtención del token de acceso.

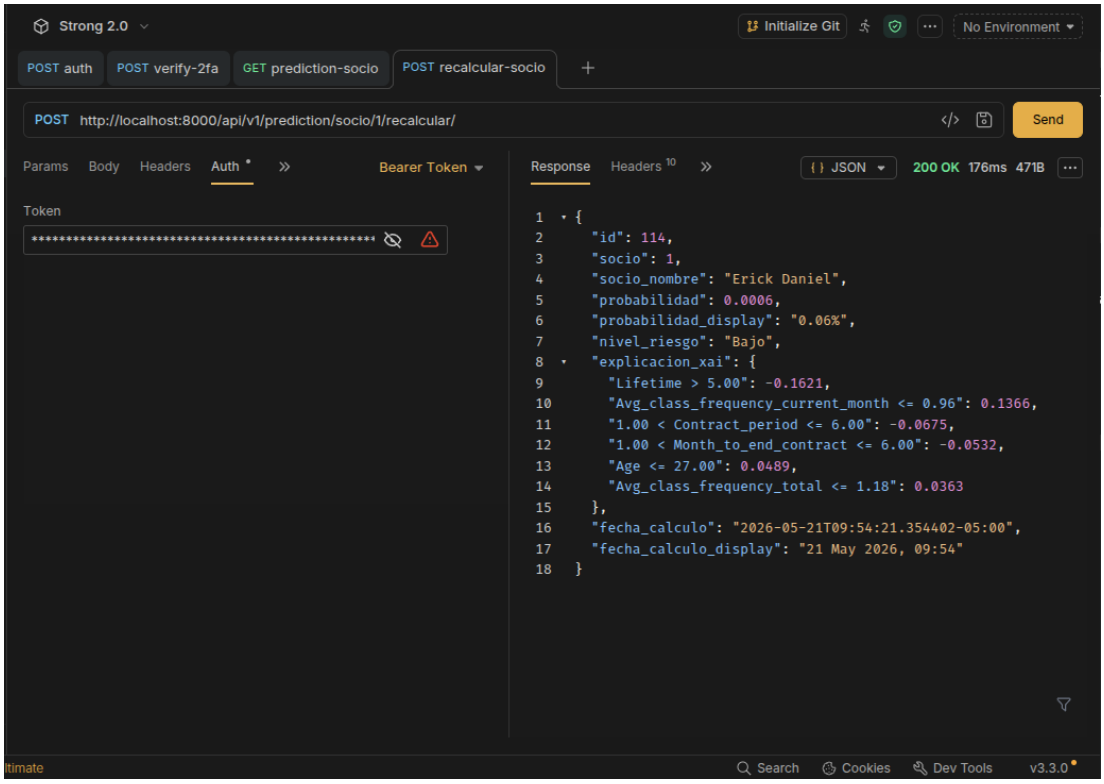


Figura 46: Prueba de integración: recálculo de predicción de un socio.

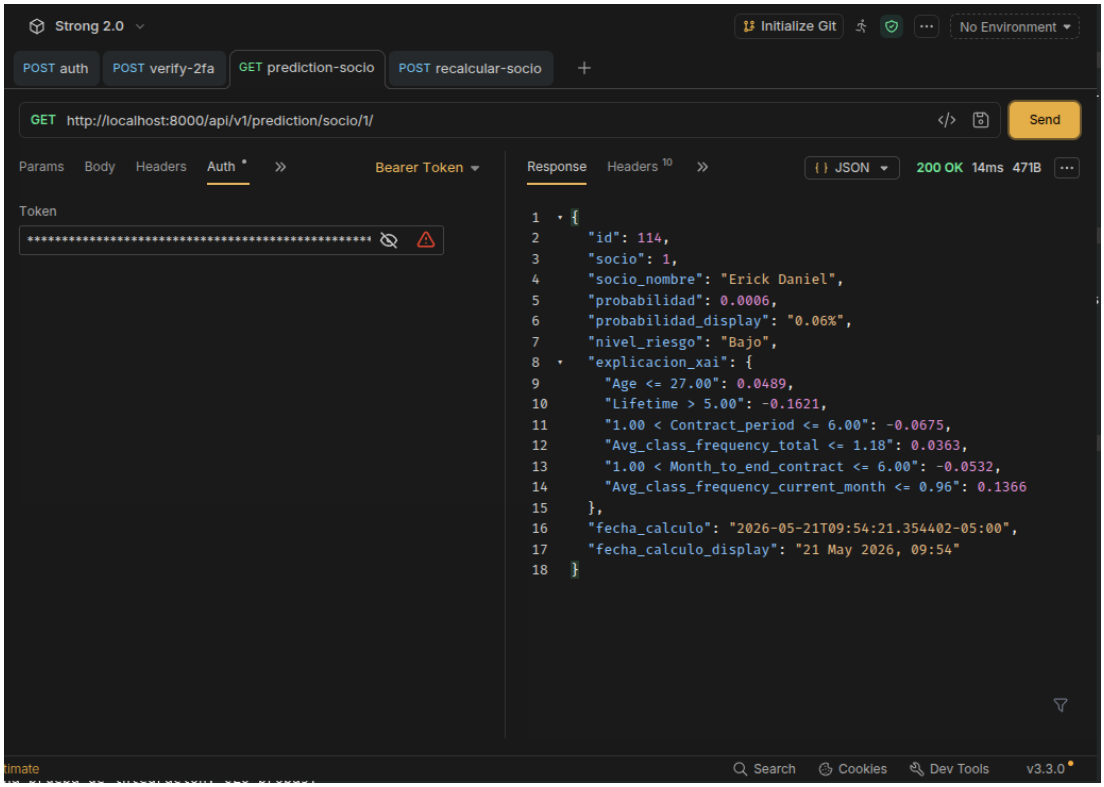


Figura 47: Prueba de integración: consulta de predicción con respuesta JSONB desde PostgreSQL.

Pruebas de caja blanca: Con *Coverage.py*, se examinaron los caminos lógicos internos del código, analizando el flujo de los middlewares de autenticación y los scripts de preprocesamiento. El objetivo fue garantizar que todas las ramas de decisión y filtros de seguridad se ejecuten sin excepciones no controladas. Los reportes de cobertura se presentan en las Figuras 48, 49 y 50.

File	statements	missing	excluded	coverage
analytics / urls.py	3	0	0	100%
analytics / views.py	135	117	0	13%
attendance / __init__.py	0	0	0	100%
attendance / admin.py	8	0	0	100%
attendance / apps.py	4	0	0	100%
attendance / migrations / __init__.py	0	0	0	100%
attendance / migrations / 0001_initial.py	6	0	0	100%
attendance / migrations / 0002_asistencia_attendance__fecha_8d9620_idx.py	4	0	0	100%
attendance / models.py	9	0	0	100%
attendance / serializers.py	8	0	0	100%
attendance / test_api.py	60	0	0	100%
attendance / tests.py	1	0	0	100%
attendance / urls.py	3	0	0	100%
attendance / views.py	117	23	0	80%
authentication / __init__.py	0	0	0	100%
authentication / admin.py	1	0	0	100%
authentication / apps.py	4	0	0	100%
authentication / migrations / __init__.py	0	0	0	100%
authentication / migrations / 0001_initial.py	8	0	0	100%
authentication / models.py	15	3	0	80%
authentication / serializers.py	16	1	0	94%
authentication / tests.py	1	0	0	100%
authentication / urls.py	4	0	0	100%
authentication / views.py	73	44	0	40%
configuration / __init__.py	0	0	0	100%
configuration / admin.py	1	0	0	100%
configuration / apps.py	4	0	0	100%
configuration / migrations / __init__.py	0	0	0	100%

Figura 48: Reporte de cobertura de código mediante Coverage.py: parte 1 de 3 (66 % de cobertura global).

configuration / migrations / 0001_initial.py	5	0	0	100%
configuration / migrations / 0002_alter_interfaceconfig_options_and_more.py	4	0	0	100%
configuration / migrations / 0003_alter_interfaceconfig_options_and_more.py	5	0	0	100%
configuration / models.py	20	0	0	100%
configuration / serializers.py	6	0	0	100%
configuration / tests.py	1	0	0	100%
configuration / urls.py	6	0	0	100%
configuration / views.py	16	8	0	50%
core / __init__.py	0	0	0	100%
core / asgi.py	4	4	0	0%
core / settings.py	40	6	0	85%
core / urls.py	7	1	0	86%
core / utils.py	13	13	0	0%
core / wsgi.py	4	4	0	0%
importar_socios.py	40	40	0	0%
manage.py	11	11	0	0%
members / __init__.py	0	0	0	100%
members / admin.py	14	0	0	100%
members / apps.py	4	0	0	100%
members / migrations / __init__.py	0	0	0	100%
members / migrations / 0001_initial.py	7	0	0	100%
members / migrations / 0002_plan_precio.py	4	0	0	100%
members / migrations / 0003_socio_cedula.py	4	0	0	100%
members / models.py	29	3	0	90%
members / serializers.py	85	7	0	92%
members / test_api.py	87	0	0	100%
members / tests.py	1	0	0	100%
members / urls.py	4	0	0	100%
members / views.py	109	29	0	73%
prediction / __init__.py	0	0	0	100%
prediction / admin.py	10	1	0	90%
prediction / apps.py	16	3	0	81%
prediction / migrations / __init__.py	0	0	0	100%
prediction / migrations / 0001_initial.py	6	0	0	100%

Figura 49: Reporte de cobertura de código mediante Coverage.py: parte 2 de 3.

prediction / migrations / 0002_ejecucionprediccion.py	6	0	0	100%
prediction / migrations / 0003_alter_prediccionchurn_options.py	4	0	0	100%
prediction / migrations / 0004_modelretrainginglog_alter_ejecucionprediccion_tipo.py	4	0	0	100%
prediction / models.py	47	3	0	94%
prediction / retraining.py	171	171	0	0%
prediction / scheduler.py	57	39	0	32%
prediction / serializers.py	34	9	0	74%
prediction / test_api.py	88	0	0	100%
prediction / test_prediction_logic.py	33	0	0	100%
prediction / test_utils.py	28	0	0	100%
prediction / tests.py	1	0	0	100%
prediction / urls.py	3	0	0	100%
prediction / utils.py	95	21	0	78%
prediction / views.py	139	35	0	75%
Total	**1757**	**596**	**0**	**66%**

Figura 50: Reporte de cobertura de código mediante Coverage.py: parte 3 de 3.

Pruebas de caja negra: Se realizaron simulaciones de usuario final para validar la respuesta del sistema ante entradas manuales. Se confirmó que la interfaz en Next.js actualiza de forma reactiva el semáforo de riesgo y los gráficos de XAI. También se validó el componente *ImpactFactors*, asegurando que la agregación de importancia de variables genere hallazgos automáticos coherentes con la situación del socio, como se aprecia en las Figuras 51, 52 y 53.

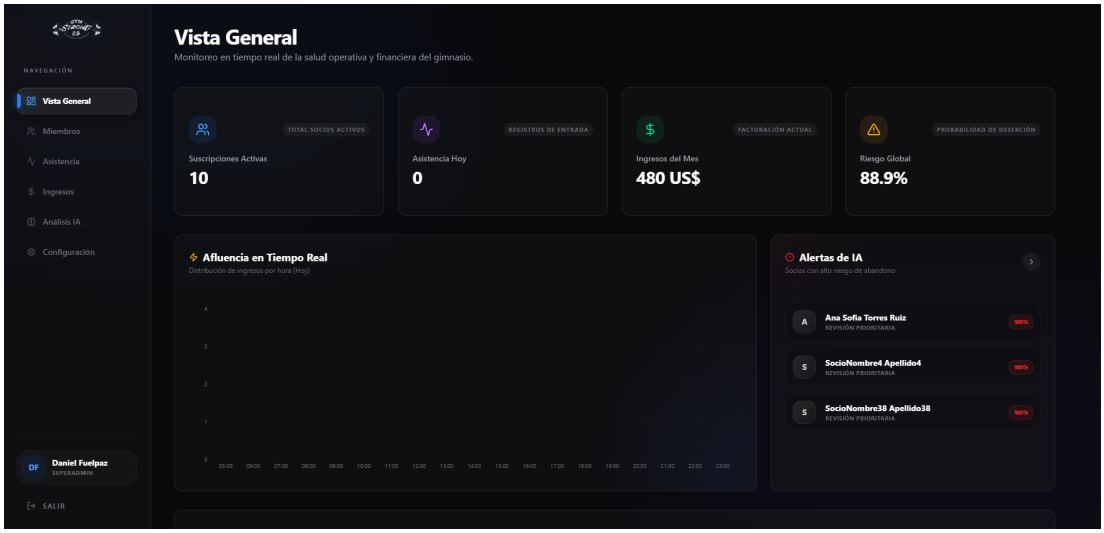


Figura 51: Prueba de caja negra: dashboard con KPI de riesgo global y alertas de socios en riesgo alto.

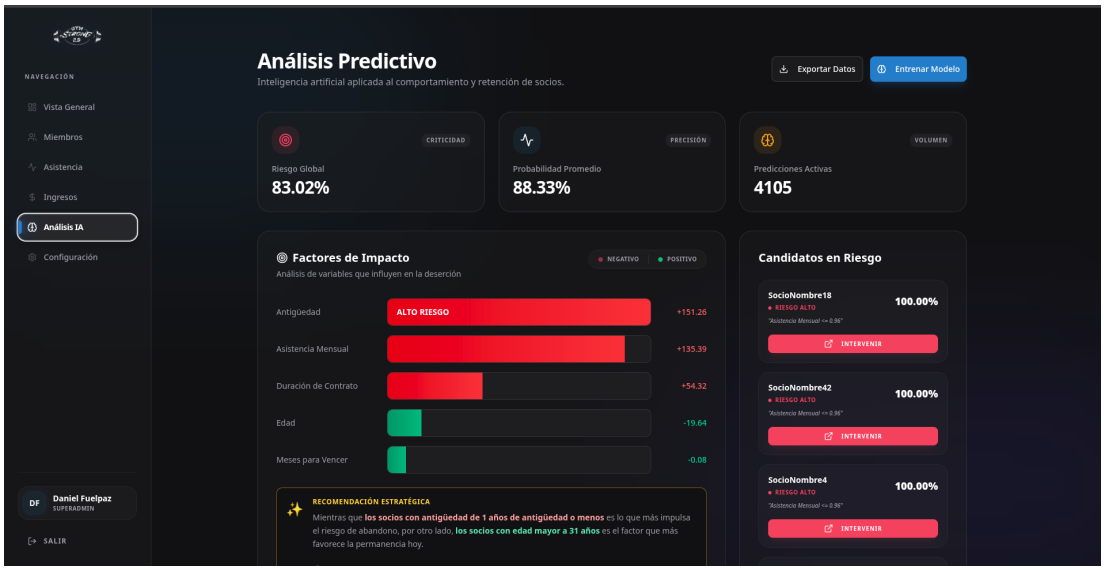


Figura 52: Prueba de caja negra: módulo de análisis predictivo con factores de impacto y candidatos en riesgo.

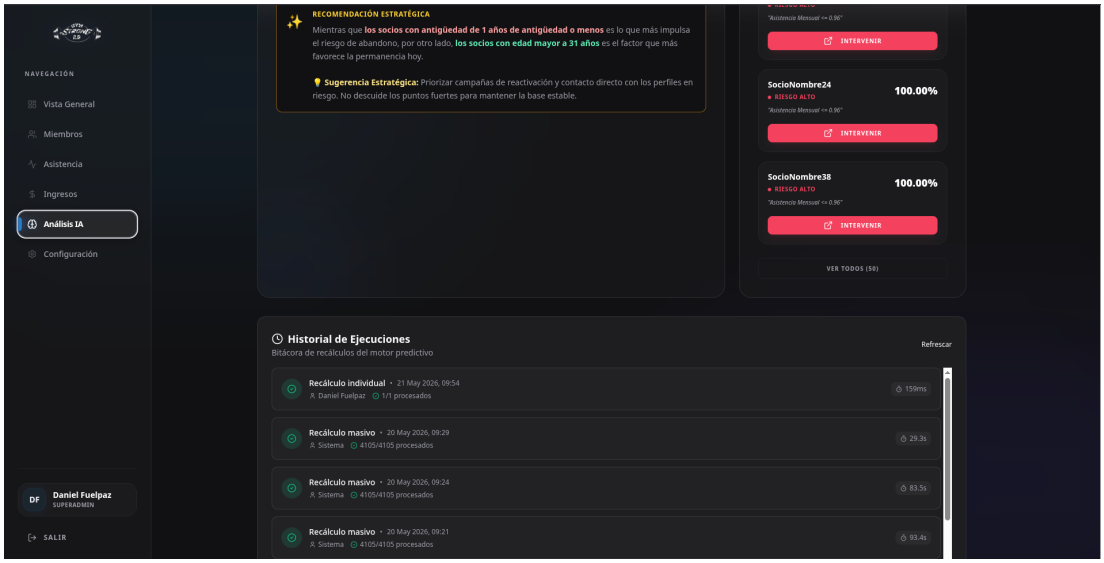


Figura 53: Prueba de caja negra: recomendación estratégica generada e historial de ejecuciones del motor predictivo.

3.4 Resultados

El análisis del gimnasio Strong 2.0 permitió identificar cuatro procesos manuales críticos: registro de socios, control de asistencia, seguimiento de membresías y cobro de mensualidades. Ninguno de ellos dejaba rastro digital confiable, lo que impedía detectar a tiempo cuándo un socio estaba por abandonar. A partir de ese diagnóstico, se establecieron 6 requisitos funcionales y 4 requisitos no funcionales, detallados en las Tablas 34 y 35 respectivamente.

Gradient Boosting alcanzó una exactitud del 93.48 % y un ROC AUC de 0.9798, los valores más altos entre los modelos evaluados, según la Tabla 48.

Tabla 48: Evaluación comparativa de modelos de Machine Learning.

Algoritmo	**Accuracy**	**Recall**	**F1-Score**	**ROC AUC**
Regresión Logística	0.9240	0.8265	0.8521	0.9729
Random Forest	0.9248	0.8341	0.8547	0.9725
SVM	0.9293	0.8313	0.8617	0.9676
Gradient Boosting	**0.9348**	**0.8492**	**0.8734**	**0.9798**

Se muestran los cuatro modelos con mejor desempeño; KNN y Árbol de Decisión se omiten por obtener los valores más bajos en todas las métricas (ver Tabla 37 para la

comparación completa). Las Figuras 54 y 55 confirman la capacidad de discriminación del modelo y el bajo índice de casos no detectados.

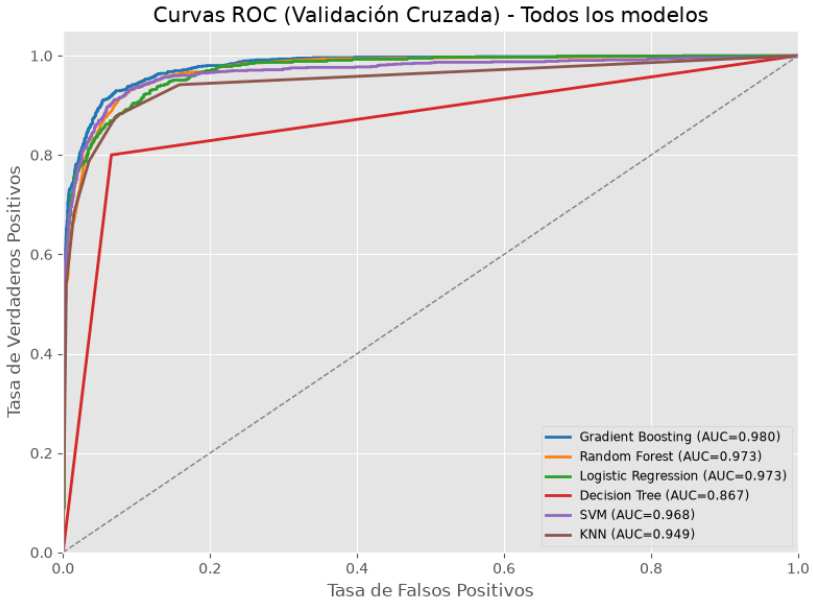


Figura 54: Análisis de discriminación del modelo mediante la Curva ROC-AUC.

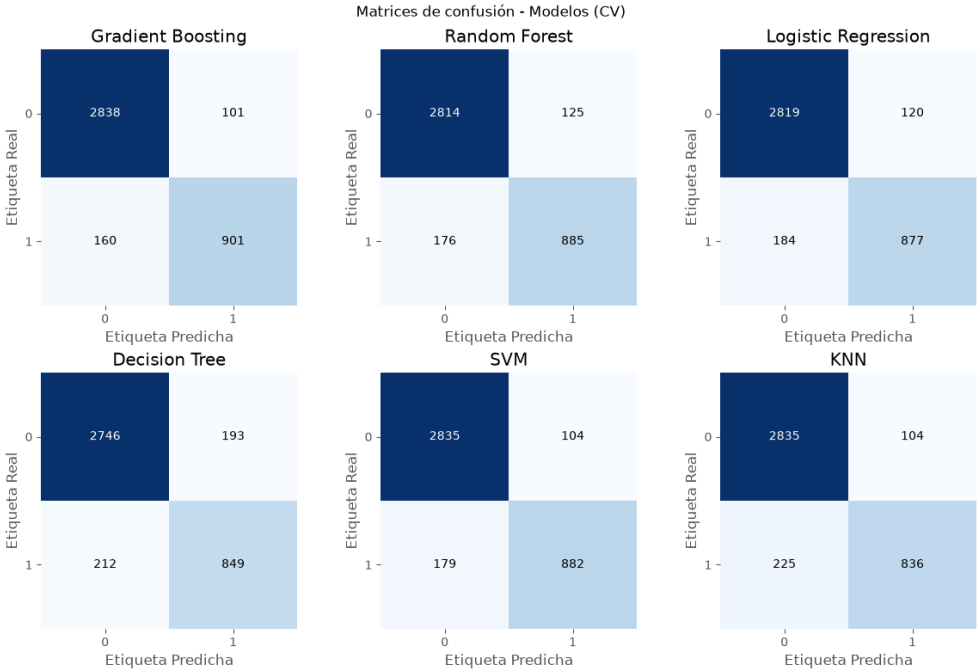


Figura 55: Matriz de Confusión: Validación de aciertos y errores predictivos.

La diferencia más relevante entre modelos se observó en la sensibilidad: Gradient Boosting superó a los demás con un 84.92 % y detectó más socios en riesgo antes de su

abandono.

Los valores de la gestión manual se obtuvieron de la entrevista realizada al administrador del gimnasio, mientras que los del sistema Strong 2.0 procedieron de las mediciones registradas durante su implementación. La Tabla 49 muestra el impacto de la automatización.

Tabla 49: Síntesis de impacto operativo: Diagnóstico vs. Implementación.

Indicador	**Gestión Manual**	**Sistema Strong 2.0**	**Mejora**
Integridad de datos	15 errores semanales	Prevenidos por centralización en BD única con validaciones de sistema	**Crítica**
Seguimiento de procesos	Sin registro digital	100 % digitalizado	**Crítica**
Reportes de socios	Manual / periódico	Dashboard en tiempo real	**Crítica**
Alertas de deserción	Inexistente	Automatizadas con justificación	**Crítica**
Actualización de predicciones	Inexistente	Automática (configurable)	**Crítica**

La integración de LIME complementó la predicción con una explicación individual por socio. Cada resultado indicó qué factores influyeron y en qué dirección, tal como se aprecia en la Figura 56.

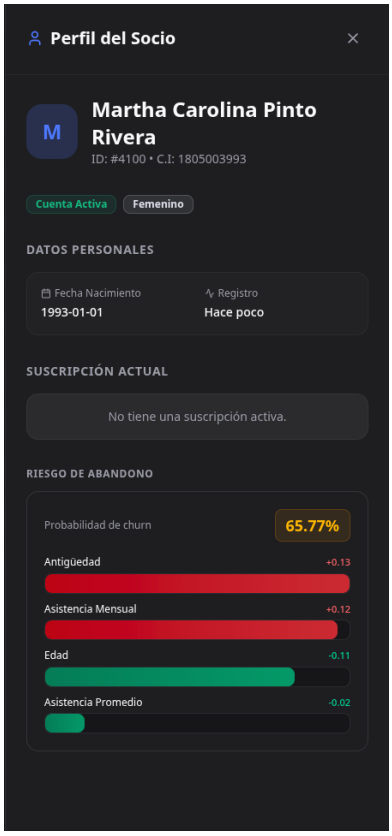


Figura 56: Explicación individual de riesgo de abandono generada por LIME para un socio.

La Figura 57 muestra el sistema en operación con datos reales de socios del gimnasio.

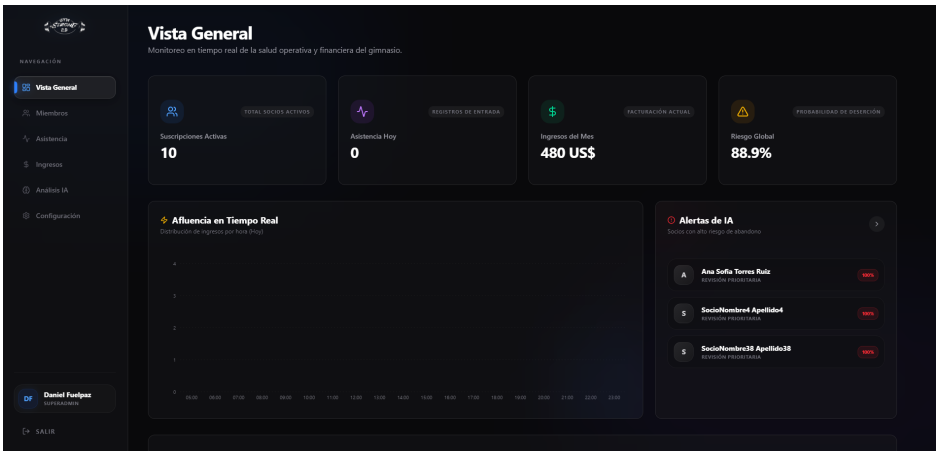


Figura 57: Predicción del modelo con datos reales de socios del gimnasio Strong 2.0.

La Tabla 50 muestra que el sistema cumplió el 100 % de los 10 requisitos planificados, tanto funcionales como no funcionales. El cumplimiento de RNF-01 se sustenta en

cinco capas de seguridad implementadas en el backend Django, cuya configuración se recoge en el Código 4:

- **Contraseñas con hash**: el modelo `User` nativo de Django aplica PBKDF2 con SHA-256 y sal aleatoria antes de persistir cualquier credencial, garantizando que las contraseñas nunca se almacenen en texto plano (comportamiento predeterminado de Django 5.2, activado por `django.contrib.auth`).
- **Autenticación JWT + 2FA**: los tokens de acceso tienen vigencia de 60 minutos; los tokens de refresco se invalidan automáticamente tras su rotación mediante `token_blacklist`. El inicio de sesión exige además la verificación de un código de seis dígitos enviado por correo electrónico con caducidad de 5 minutos (`TwoFactorCode`).
- **Control de acceso por roles**: todos los *endpoints* requieren autenticación (`IsAuthenticated` como política global); el acceso a módulos sensibles se restringe adicionalmente por rol de usuario.
- **Protección del transporte**: en producción se activan HSTS con 365 días (`SECURE_HSTS_SECONDS = 31 536 000`), `SESSION_COOKIE_SECURE` y `CSRF_COOKIE_SECURE`, asegurando que toda la comunicación se curse sobre HTTPS.
- **Trazabilidad de cambios**: `django-simple-history` registra automáticamente cada modificación sobre `Socio` y `Suscripcion`, generando un historial de auditoría que permite identificar alteraciones no autorizadas.
- **Enmascaramiento de datos personales**: la función `mask_data()` en la capa de serialización entrega versiones parciales de cédula y nombre según el rol del solicitante. El personal administrativo recibe `172******12` y `Eri**********az`; el superusuario recibe el dato completo. Los campos originales (`cedula`, `nombres`, `apellidos`) se declaran `write_only` en el serializador, de modo que nunca se exponen en el JSON de respuesta.

```
# Validacion de contrasenas (4 reglas activas)
AUTH_PASSWORD_VALIDATORS = [
    {'NAME': '...UserAttributeSimilarityValidator'},
    {'NAME': '...MinimumLengthValidator'},
    {'NAME': '...CommonPasswordValidator'},
    {'NAME': '...NumericPasswordValidator'},
]

# JWT: 60 min de acceso, rotacion con blacklist
SIMPLE_JWT = {
    'ACCESS_TOKEN_LIFETIME': timedelta(minutes=60),
    'REFRESH_TOKEN_LIFETIME': timedelta(days=1),
    'ROTATE_REFRESH_TOKENS': True,
    'BLACKLIST_AFTER_ROTATION': True,
}

# Seguridad en produccion
SECURE_HSTS_SECONDS = 31536000
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
```

Código 4: Configuración de seguridad en `settings.py` (extracto).

Tabla 50: Cumplimiento de requisitos funcionales y no funcionales.

ID	Nombre	Estado
RF-01	Gestión de socios	✓
RF-02	Control de membresías	✓
RF-03	Registro de asistencia	✓
RF-04	Motor de predicción	✓
RF-05	Alertas de riesgo	✓
RF-06	Explicación de la IA	✓
Total Funcionales		**6 de 6**
RNF-01	Seguridad e integridad	✓
RNF-02	Mantenibilidad y estructura	✓
RNF-03	Compatibilidad de datos	✓
RNF-04	Exactitud analítica	✓
Total No Funcionales		**4 de 4**

3.5 Discusión

Los resultados se contrastan con trabajos similares de la región. En cuanto a tecnología, tanto Bolaños Hidalgo [2] como Pazmiño Rodríguez [3] adoptaron stacks convencionales, Java/Spring/MySQL y Next.js/Node.js respectivamente, que satisfacen los requisitos de gestión operativa básica. La plataforma de Enríquez Vinces [4] optó por Django como único framework, sin separación entre capa de presentación y lógica de negocio. Strong 2.0, en cambio, combina Django como API REST con Next.js como frontend desacoplado, lo que permite escalar ambas capas de forma independiente y ofrece una arquitectura más sólida para integrar componentes de análisis de datos.

El aporte más relevante de este trabajo es la incorporación de un motor de inteligencia artificial. Ninguno de los trabajos comparados incluye mecanismo alguno de predicción o análisis avanzado de datos de socios: todos limitan su alcance a la digitalización de procesos administrativos. Strong 2.0 supera ese umbral al integrar un modelo *Gradient Boosting* con un ROC AUC de 0.98 y una exactitud del 93.48 %, capaz de identificar al 85 % de los casos positivos en el conjunto de validación. LIME transforma esa predicción en una explicación interpretable e individual por socio, dimensión que ningún trabajo previo del dominio local contempla.

En cuanto a métricas de calidad, el trabajo de Bolaños Hidalgo [2] es el único que reporta un indicador cuantitativo de eficiencia de desempeño bajo ISO/IEC 25010, alcanzando un 81.94 % mediante la reducción del tiempo de respuesta en un 52.77 %. Strong 2.0 no reporta una métrica equivalente, pero alcanzó el 66 % de cobertura de código en el backend y cumplió el 100 % de los 10 requisitos planificados.

Estos resultados tienen limitaciones concretas. El modelo predictivo fue entrenado con datos históricos de un único gimnasio de tamaño mediano, lo que restringe la generalización a establecimientos con perfiles de socios muy diferentes. La capacidad predictiva podría degradarse en contextos donde las variables de asistencia o pago sigan patrones diferentes. Ampliar el conjunto de datos con información de múltiples centros deportivos o implementar estrategias de reentrenamiento periódico constituye la principal vía de mejora identificada.

Este trabajo aporta al estado del arte regional un sistema que integra, por primera vez en este dominio, gestión operativa completa con predicción explicable de deserción. Mientras los trabajos precedentes resuelven el problema de la digitalización, Strong 2.0 aborda un problema más complejo: anticipar el comportamiento del socio y dotar al administrador de evidencia concreta para actuar. Este trabajo establece un precedente para futuros sistemas de gestión deportiva que integren inteligencia artificial con capacidad explicativa.

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se identificaron los procesos manuales del gimnasio Strong 2.0 mediante un diagnóstico inicial que evidenció la ausencia de trazabilidad digital en las operaciones de registro, asistencia, membresías y cobros. A partir de ese diagnóstico se definieron los requisitos funcionales y no funcionales que guiaron el desarrollo del sistema, sintetizados en las Tablas 34 y 35.

- Se seleccionó Gradient Boosting como modelo predictivo de deserción a partir del análisis comparativo de la Tabla 37, alcanzando una exactitud del 93,48 %, sensibilidad del 85 % y ROC AUC de 0,98. Para resolver la baja interpretabilidad propia del algoritmo, se integró LIME como técnica de Inteligencia Artificial Explicable, lo que permite al administrador identificar qué factores pesan más en el riesgo de cada socio y actuar con decisiones fundamentadas en evidencia, como muestra la Figura 56.

- Se implementó el sistema web Strong 2.0 con Django como API REST y Next.js como interfaz desacoplada, según la arquitectura ilustrada en la Figura 29 y la selección tecnológica fundamentada en las Tablas 44 y 45. El sistema centralizó la gestión administrativa e integró el modelo predictivo en un único flujo de trabajo. Los diseños de interfaz que guiaron el desarrollo se presentan en los Anexos A, B, C, D y E. El sistema cumplió el 100 % de los requisitos planificados y alcanzó una cobertura de código del 66 % en el backend, verificada en las Figuras 48 al 50.

- El sistema web desarrollado para el gimnasio Strong 2.0 durante el período académico Enero–julio 2026 centralizó la gestión de suscripciones y asistencia, digitalizó los cuatro procesos manuales identificados e integró un modelo predictivo de deserción con exactitud del 93,55 %, complementado con explicabilidad individual mediante LIME. El resultado es una herramienta operativa y analítica que no existía previamente en el establecimiento y que responde directamente a la problemática diagnosticada.

4.2 Recomendaciones

- Se recomienda integrar hardware de control de acceso biométrico o por tarjeta RFID con la API REST del sistema para eliminar la intervención manual en el registro de asistencia y mejorar la precisión de los datos que alimentan al modelo predictivo.

- Se recomienda ampliar el módulo de inteligencia artificial hacia la proyección financiera, incorporando modelos de series temporales para anticipar los ingresos mensuales a partir de las tendencias de renovación y deserción que el sistema ya identifica.

- Se recomienda replicar la metodología empleada en esta investigación en otros centros deportivos o negocios basados en suscripciones, con el fin de validar su capacidad de generalización más allá del contexto de Strong 2.0.

- Se recomienda comparar el desempeño de LIME con otras técnicas de explicabilidad como SHAP en escenarios similares de predicción de deserción, para determinar cuál ofrece mayor fidelidad y utilidad práctica para el administrador.

- Se recomienda incorporar la temática de predicción de deserción mediante aprendizaje automático como contenido de estudio en las carreras de Software,Tecnologías de la Información, y extenderla de forma transversal a facultades donde la gestión de clientes sea relevante, como Administración de Empresas o Ciencias Económicas.

- Se recomienda profundizar en la relación entre factores vinculados a la formación de hábitos y las variables predictoras del modelo, como la frecuencia de asistencia o la duración de membresía, con el fin de ampliar la capacidad explicativa del sistema en futuros estudios.

REFERENCIAS BIBLIOGRÁFICAS

[1] A. Vinueza, "Gym customers features and churn." https://www.kaggle.com/datasets/adrianvinueza/gym-customers-features-and-churn, 2024. Dataset público disponible en Kaggle.

[2] P. S. Bolaños Hidalgo, "Desarrollo de una aplicación web para la gestión de clientes y control de pagos de mensualidades del gimnasio "team legión" riobamba," Master's thesis, Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador, 2023.

[3] F. A. Pazmiño Rodríguez, "Desarrollo de un sistema web y aplicación móvil para la gestión operativa de un centro de crossfit de la ciudad de quito," Master's thesis, Escuela Politécnica Nacional, Escuela de Formación de Tecnólogos, Quito, Ecuador, agosto 2024.

[4] A. A. Enríquez Vinces, "Plataforma de gestión para gimnasio con registro de suscripciones, en el cantón pindal de la provincia de loja en el periodo comprendido de 2022 - 2023," Master's thesis, Instituto Tecnológico Superior "Los Andes", Loja, Ecuador, 2023.

[5] J. M. Ayala Santana and A. M. Palta Naula, "Desarrollo de una aplicación web para controlar las actividades de un centro de entrenamiento deportivo," proyecto técnico de ingeniería en sistemas computacionales, Universidad Estatal de Milagro, Facultad de Ciencias e Ingeniería, Milagro, Ecuador, julio 2021.

[6] M. Morán, "Establecer un sistema para gestionar procesos para la empresa station 24 fitness implementando un primer proceso basado en la medida de satisfacción del cliente," Master's thesis, Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO), Tlaquepaque, Jalisco, México, mayo 2023.

[7] C. A. Mier Cadena, "Desarrollo de un sistema web de gestión de base de datos de tipo gerencial para la optimización financiera de gimnasios," Master's thesis, Es-

cuela Politécnica Nacional, Escuela de Formación de Tecnólogos, Quito, Ecuador, marzo 2024.

[8] M. P. Vargas Salazar and A. D. Granda Rivera, "Desarrollo de un sistema web en plataforma .net y una aplicación móvil nativa android para la gestión de los gimnasios de la ciudad de quito. caso de aplicación gimnasio anahí," Master's thesis, Escuela Politécnica Nacional, Quito, Ecuador, junio 2016.

[9] D. E. Fajardo Rubio, "Aplicación web para el manejo de las actividades administrativas y operativas del gimnasio hammer gym del cantón naranjal," Master's thesis, Universidad Agraria del Ecuador, Facultad de Ciencias Agrarias, Carrera de Ingeniería en Computación e Informática, Milagro, Ecuador, julio 2020.

[10] P. Sobreiro, P. Guedes-Carvalho, A. Santos, P. Pinheiro, and C. Gonçalves, "Predicting fitness centre dropout," *International Journal of Environmental Research and Public Health*, vol. 18, 10 2021.

[11] M. Adrin and D. Fadaralika, "A comparative model for predicting customer churn using supervised machine learning," *International Journal of Science and Research (IJSR)*, vol. 11, pp. 133–136, 2 2022.

[12] A. Manzoor, M. A. Qureshi, E. Kidney, and L. Longo, "A review on machine learning methods for customer churn prediction and recommendations for business practitioners," *IEEE Access*, vol. 12, pp. 70434–70463, 2024.

[13] M. Aldosary and A. Alrashdan, "Churn prediction for gym members using artificial neural networks assisted with the psychological concept of habit formation in the fitness industry," tech. rep., 2021.

[14] X. Yu, L. Hu, B. Reichman, D. Chu, R. Chandrupatla, X. Zhang, L. Heck, and T. Rosing, "Sensorchat: Answering qualitative and quantitative questions during long-term multimodal sensor interactions," 5 2025.

[15] B. Amiri and S. H. Hosseini, "Unveiling the power of social influence: A machine learning framework for churn prediction with network analysis," *IEEE Access*, vol. 12, pp. 71271–71285, 2024.

[16] B. Papis, K. Grochowski, K. Subzda, and K. Sijko, "Experimental evaluation of test-driven development with interns working on a real industrial project," *IEEE Transactions on Software Engineering*, vol. 48, pp. 1644–1664, 5 2022.

[17] N. Pombo and C. Martins, "Test driven development in action: Case study of a cross-platform web application," in *EUROCON 2021 - 19th IEEE International Conference on Smart Technologies, Proceedings*, pp. 352–356, Institute of Electrical and Electronics Engineers Inc., 7 2021.

[18] J. Zhu, A. Guha, D. Do, M. Xu, X. Nguyen, D. Zhao, and T. Jaakkola, "Functional optimal transport: regularized map estimation and domain adaptation for functional data," tech. rep., 2024.

[19] U. Bhatnaggar, M. Dubey, U. S. Bhatnagar, and A. Pandey, "Customer churn prediction using machine learning classifiers," *International Journal of Research Publication and Reviews*, vol. 5, pp. 7588–7594, 1 2024.

[20] B. Zhang, "Customer churn in subscription business model-predictive analytics on customer churn," tech. rep., 2023.

[21] M. Maduna, A. Telukdarie, I. Munien, U. Onkonkwo, and A. Vermeulen, "Smart customer churn management system using machine learning," in *Procedia Computer Science*, vol. 237, pp. 552–558, Elsevier B.V., 2024.

[22] D. Ankunda, G. Marvin, and N. Kimbugwe, "Local interpretable model-agnostic approaches to gym crowd predictive modeling with ensemble learning," 2024.

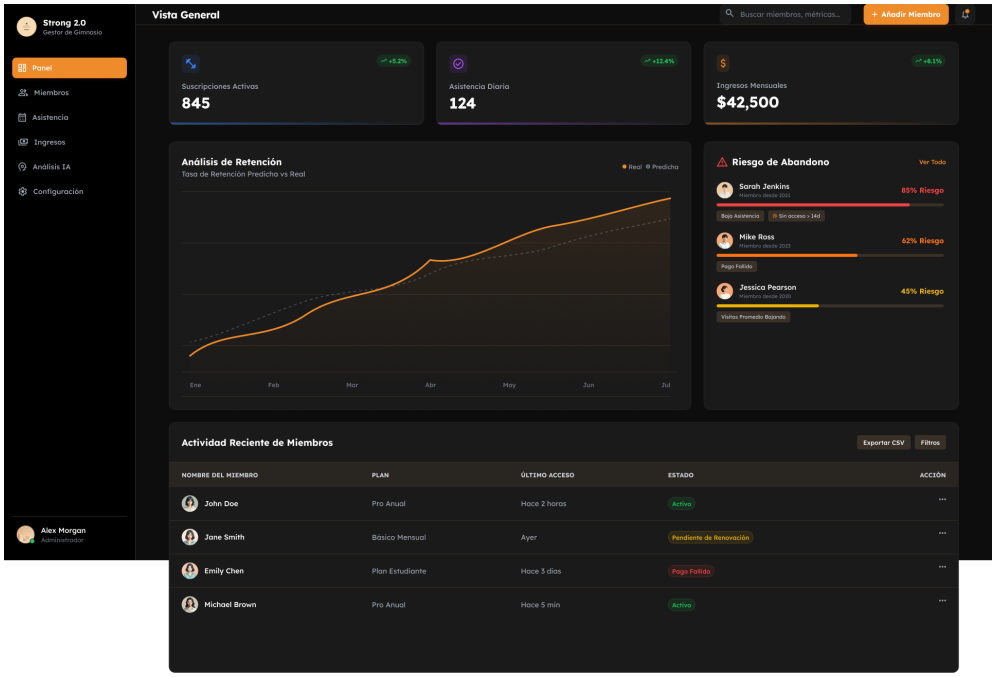
[23] P. A. Lavy Aguilar and A. B. Rios Cotrina, "Impacto de un aplicativo software en la gestión de clientes de un gimnasio, cajamarca 2023," Master's thesis, Universidad Nacional de Cajamarca, Facultad de Ingeniería, Carrera de Ingeniería de Sistemas Computacionales, Cajamarca, Perú, 2023.

[24] W. Wang, S. Nepal, J. F. Huckins, L. Hernandez, V. Vojdanovski, D. Mack, J. Plomp, A. Pillai, M. Obuchi, A. Dasilva, E. Murphy, E. Hedlund, C. Rogers, M. Meyer, and A. Campbell, "First-gen lens: Assessing mental health of first-generation students across their first year at college using mobile sensing," *Procee-*

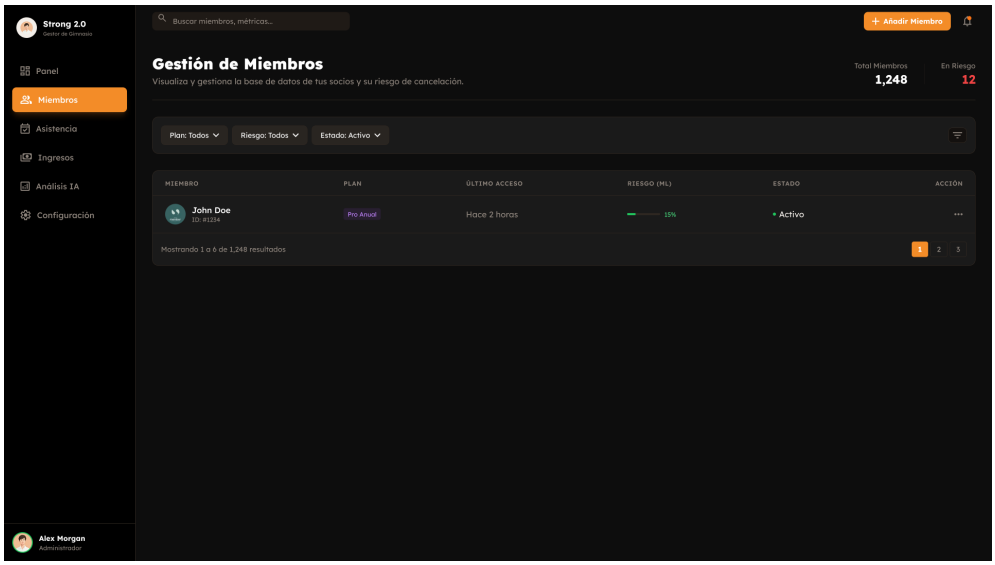
dings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 6, 7 2022.

[25] O. K. Richards and T. C. Veinot, "Reconceptualizing technology for chronic disease management activities in the family: Supporting collective routines," *ACM Transactions on Computer-Human Interaction*, vol. 32, 6 2025.

ANEXOS

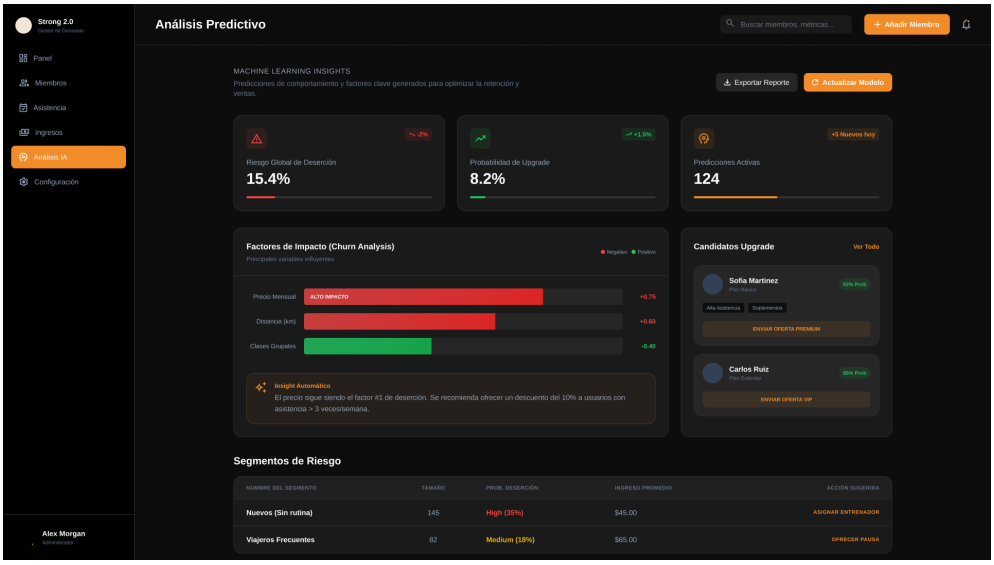


Anexo A. Mockup del dashboard de gestión.



Anexo B. Mockup de la pantalla de gestión de miembros.

Anexo C. Mockup de la pantalla de configuración del sistema.



Anexo D. Mockup de la pantalla de análisis de inteligencia artificial.

Anexo E. Mockup de la pantalla de alertas de riesgo.

Anexo F. Especificación de casos de uso complementarios.

Las Tablas 51, 52 y 53 detallan la especificación de los casos de uso complementarios del sistema Strong 2.0.

Tabla 51: Especificación CU-04: Gestionar Socio.

Actor	Administrador / Entrenador–Staff
Precondición	El usuario ha iniciado sesión con permisos de administrador o staff.
Flujo principal	1. El administrador accede al módulo de gestión de socios. 2. El sistema muestra el listado de socios con filtros de búsqueda. 3. El administrador selecciona la opción de crear nuevo socio. 4. El sistema presenta el formulario con los campos: nombre, apellido, cédula, fecha de nacimiento, correo y teléfono. 5. El administrador completa el formulario y selecciona el plan de membresía. 6. El sistema valida los datos, crea el perfil del socio y asigna la membresía. 7. El sistema confirma el registro y muestra el perfil creado.
Flujo alternativo	3a. El administrador selecciona un socio existente para editar su información: el sistema carga el formulario con los datos actuales y persiste los cambios al guardar. 6a. Cédula duplicada: el sistema notifica el conflicto y solicita corrección antes de continuar.
Postcondición	El perfil del socio queda registrado o actualizado en la base de datos.

Tabla 52: Especificación CU-05: Gestionar Membresía.

Actor	Administrador
Precondición	El socio tiene un perfil registrado en el sistema.
Flujo principal	1. El administrador accede al perfil del socio o al módulo de membresías. 2. El sistema muestra el estado actual: plan, fecha de inicio, fecha de vencimiento y estado de pago. 3. El administrador registra el pago de la mensualidad o renueva el plan. 4. El sistema actualiza la fecha de vencimiento y registra el pago. 5. El sistema confirma la operación y establece el estado del socio como activo.
Flujo alternativo	3a. El administrador cambia el plan de membresía: el sistema recalcula la fecha de vencimiento según el nuevo plan. 4a. Pago registrado fuera del período de gracia: el sistema aplica el parámetro `dias_gracia` configurado antes de marcar al socio como inactivo.
Postcondición	La membresía del socio queda actualizada con el nuevo estado de pago y fechas correspondientes.

Tabla 53: Especificación CU-06: Gestionar Alertas de Riesgo.

Actor	Administrador
Precondición	El sistema ha ejecutado al menos una inferencia (incluye CU-02).
Flujo principal	1. El administrador accede al módulo de alertas desde el panel. 2. El sistema muestra el listado de socios clasificados por nivel de riesgo: Alto, Medio y Bajo. 3. El administrador selecciona un socio para ver el detalle de la alerta. 4. El sistema presenta la probabilidad de deserción, la clasificación de riesgo y los factores explicativos generados por LIME (incluye CU-03). 5. El administrador identifica los socios en riesgo alto para diseñar acciones de retención.
Flujo alternativo	2a. Sin socios en riesgo alto: el sistema muestra el panel con socios en riesgo bajo o medio. 4a. La alerta fue generada por el umbral de inactividad manual (`dias_inactividad_manual`): el sistema la presenta diferenciada de las alertas del modelo predictivo.
Postcondición	El administrador cuenta con la información necesaria para implementar estrategias de retención focalizadas.