'Radar Premios' — Arquitectura y Plan Estratégico para un Sistema de Pronósticos Riguroso

1.0 Resumen Ejecutivo

El presente informe técnico detalla una propuesta de arquitectura lógica y un plan de acción para la evolución del sistema 'Radar Premios'. El análisis del documento inicial [1] revela un sistema con fundamentos teóricos sólidos, un enfoque metodológico riguroso y una estructura de datos bien definida. Sin embargo, su dependencia de mecanismos de orquestación como

schtasks para la ejecución de scripts en un entorno local introduce fragilidades operativas, como la falta de escalabilidad, la dificultad de monitoreo centralizado y la vulnerabilidad ante fallos de hardware o software.[2]

Para mitigar estos desafíos, se propone una modernización integral hacia una arquitectura de microservicios y pipelines de datos. Esta arquitectura desacopla las funcionalidades críticas del sistema (ingesta de datos, cálculo de señales, generación de candidatos, evaluación y reportes) en servicios autónomos, lo que potencia la resiliencia y facilita la gestión de errores.[3] La orquestación de este flujo de trabajo se gestionará mediante una plataforma de código abierto como Apache Airflow [5], reemplazando la programación de tareas rígida y monolítica.

La visión estratégica es transformar 'Radar Premios' de una colección de scripts en una plataforma de software auditable, trazable y lista para la producción. El plan de acción se estructura en tres fases. La primera, enfocada en la modernización del entorno con Docker y Apache Airflow. La segunda, en el fortalecimiento del rigor científico mediante la implementación de modelos de Markov avanzados y el versionado de datos con Data Version Control (DVC). Finalmente, la tercera fase se centra en la optimización y el monitoreo avanzado del sistema en producción, incluyendo la detección de deriva de datos y la automatización del ajuste de modelos. Este enfoque metódico garantiza que el proyecto no solo aborde sus desafíos inmediatos, sino que también establezca un marco robusto y

sostenible para su futuro crecimiento.

2.0 Fundamentos Conceptuales y Filosóficos del Sistema

2.1 Del Caos al Orden: El Legado de la Teoría de la Probabilidad

La filosofía del sistema 'Radar Premios' se cimenta en los principios de la teoría de la probabilidad, negando la idea de que los juegos de azar son un dominio de caos puro. Por el contrario, se basan en la premisa de que incluso los eventos dependientes pueden ser modelados y analizados. Esta perspectiva encuentra sus raíces en una de las disputas científicas más influyentes de la historia de la matemática.

En 1905, en la Rusia zarista, Andrey Markov, un ateo riguroso, se opuso a Pavel Necrasov, un matemático y religioso devoto que creía que la Ley de los Grandes Números solo se aplicaba a eventos independientes, lo que a su vez probaba la existencia del libre albedrío en las estadísticas sociales como la tasa de matrimonios o crímenes.[1] Markov, en un esfuerzo por refutar esta postura, se propuso demostrar que los eventos dependientes también pueden seguir la Ley de los Grandes Números.[1] Para ello, ideó un sistema que analizaba la secuencia de vocales y consonantes en un poema, demostrando que la probabilidad de la siguiente letra dependía de la letra actual. Creó un modelo de transición de estados que, con suficientes repeticiones, convergía a una distribución estable de vocales y consonantes, similar a la observada en el poema original.[1] Con este sistema, conocido como Cadena de Markov, encontró una forma de hacer cálculos de probabilidad con eventos dependientes. El sistema 'Radar Premios' adopta esta visión. En lugar de asumir que cada sorteo es un evento completamente aleatorio e independiente, busca patrones de dependencia en la historia de los resultados, como la probabilidad de que un dígito aparezca después de otro en una posición específica. El sistema utiliza matrices de transición de Markov para modelar esta "memoria de primer orden" entre los dígitos y signos, lo que constituye uno de sus pilares fundamentales.[1]

Este enfoque se alinea con la visión de Andrey Kolmogorov, quien en 1933 transformó la teoría de la probabilidad de un conjunto de fórmulas dispersas a una ciencia exacta, basada en axiomas firmes. Para Kolmogorov, el azar no era un caos sin ley, sino un territorio que podía ser estructurado y descrito matemáticamente. Su trabajo sentó las bases para que la

probabilidad se convirtiera en un lenguaje universal capaz de modelar el clima, la mecánica cuántica o la economía, elevándola a la altura de las ramas más nobles de la ciencia. La aplicación de este rigor a un dominio como los juegos de azar es un reflejo de este legado: 'Radar Premios' no busca "adivinar" resultados, sino aplicar principios científicos para analizar y modelar la incertidumbre de una manera medible y rigurosa, siguiendo la tradición de convertir lo incierto en una ciencia.

2.2 La Paradoja del Azar y el Imperativo de la Reproducibilidad

La solidez de 'Radar Premios' no solo reside en la aplicación de modelos estadísticos, sino en su adhesión a un principio filosófico crucial: la **reproducibilidad constructiva**. El sistema se posiciona en contra de las "elecciones arbitrarias" o no medibles. Esto se inspira en el Axioma de Elección de la teoría de conjuntos, el cual establece que es posible seleccionar un elemento de un número infinito de conjuntos no vacíos, incluso si no existe una regla clara o "constructiva" para hacerlo. Si bien este axioma es útil para ciertas demostraciones matemáticas, también conduce a paradojas contraintuitivas, como la de Vitali, que permite construir un conjunto de números sin una longitud consistente, o la de Banach-Tarski, que teóricamente permite dividir una esfera en cinco piezas y reensamblarlas en dos esferas idénticas a la original. Estos resultados absurdos, aunque teóricamente posibles, demuestran que las construcciones no medibles pueden violar la lógica intuitiva del mundo físico y la medición.

'Radar Premios' rechaza explícitamente estas construcciones patológicas. En su lugar, se apoya en un generador de números pseudoaleatorios (PRNG) con una semilla determinista. A diferencia de la aleatoriedad verdadera, que es impredecible, la pseudoaleatoriedad es completamente reproducible. Cada corrida del algoritmo, con la misma semilla, producirá exactamente la misma secuencia de candidatos. Esta adhesión a un "buen orden constructivo" es un imperativo de la ciencia de datos: asegura que cada generación de candidatos sea auditable y que cualquier discrepancia en el rendimiento pueda ser trazada a un cambio explícito en el código, los datos o los parámetros del modelo, y no a una "elección" inexplicable.[1]

El compromiso con el rigor matemático y la transparencia es un elemento central del sistema. El material de referencia aborda la psicología del juego, mostrando cómo los jugadores a menudo se dejan llevar por el "pensamiento mágico" y la ilusión de la suerte. El sistema 'Radar Premios', al ser un ejercicio de ciencia aplicada, busca deconstruir esta ilusión. Al declarar de manera explícita que "no hay garantías de acierto" y que se deben separar "hechos de hipótesis" en los reportes [1], el sistema no solo cumple con un requisito ético de juego responsable, sino que también se posiciona como una herramienta de "desmitificación". Su valor no reside en la promesa de un premio, sino en la capacidad de demostrar que, incluso

en un dominio aparentemente regido por el azar, se puede aplicar la lógica, el análisis estadístico y la reproducibilidad. Esta postura eleva su valor más allá de una simple herramienta de pronósticos, convirtiéndola en un ejercicio de transparencia y racionalidad que promueve una relación más sana y consciente con el juego.

3.0 La Arquitectura Lógica de 'Radar Premios' 2.0

3.1 Un Salto de Fe Tecnológico: De schtasks a Microservicios y Pipelines

La arquitectura actual, basada en scripts de Python ejecutados por el programador de tareas de Windows (schtasks), es inherentemente frágil. Una falla en un script o en el entorno puede detener todo el proceso, carece de un sistema de reintentos automático robusto y dificulta la escalabilidad.[1] Para superar estas limitaciones, se propone una arquitectura de microservicios y pipelines de datos, un modelo de desarrollo de software nativo de la nube que divide una aplicación en sus componentes más pequeños e independientes.[3]

La modernización del entorno se fundamenta en la containerización. Los contenedores de Docker permiten empaquetar una aplicación y todas sus dependencias en un entorno aislado, asegurando que los scripts de Python funcionen de manera idéntica en cualquier ambiente.[10] La orquestación de estos contenedores se gestionará con Apache Airflow, un estándar de código abierto para la definición, programación y monitoreo de flujos de trabajo (

DAGs o Directed Acyclic Graphs).[5] A diferencia de la programación estática de

schtasks, un DAG de Airflow es un flujo de trabajo programable en Python, lo que permite una mayor flexibilidad para definir dependencias entre tareas, reintentos automáticos y una visibilidad completa del estado de la ejecución a través de su interfaz de usuario.[5] La migración de la lógica de programación desde un archivo

.bat (como se detalla en el documento schtasks /Create /TR "\"%MASTER% gen...\"") a la sintaxis programática de un DAG de Airflow es un paso fundamental para lograr la automatización y la resiliencia.[11]

3.2 El Corazón del Sistema: Módulos del Pipeline

El sistema se estructura en una serie de microservicios, cada uno con una función específica dentro del pipeline de datos:

- **Módulo de Ingesta (Extract):** Este servicio es responsable de la recolección de los resultados de los sorteos. Su función principal es hacer pull de los datos de las fuentes, como se describe en el documento inicial (scrapea resultados según cada sorteo[1]). La resiliencia es clave en esta etapa; por lo tanto, el módulo debe implementar una política de reintentos con
  fallback URL para gestionar fallos de conexión o errores HTTP 500.[1] A futuro, esta ingesta podría evolucionar a un sistema de
  streaming en tiempo real, utilizando tecnologías como Apache Kafka o NiFi, para procesar la información a medida que se genera.[12]
- **Módulo de Procesamiento y Generación (Transform & Generate):** Este es el núcleo del sistema de pronósticos. A partir de los datos históricos, este módulo calcula las "señales" o features. El documento original propone un conjunto de señales basado en la matemática clásica:
  - Hot/Cold: Frecuencia de aparición de dígitos o signos en ventanas de tiempo específicas.[1]
  - Last-seen: Distancia en días desde la última aparición de un dígito en una posición.[1]
  - Markov posicional: Matrices de transición t-1 -> t por dígito y posición, capturando la dependencia entre eventos.[1]
  - Patrón y Complejidad: Detección de patrones triviales (consecutivos, repetidos) o secuencias de baja complejidad.[1]

  La propuesta de modernización eleva el método de puntuación. Si bien el documento inicial sugiere una suma lineal ponderada de señales (score = \sum (w_i * s_i)) [1], una mejora significativa es la adopción de un modelo deMachine Learning (ML) para esta tarea. Un modelo como una **Red Neuronal Simple** o un **modelo de Gradient Boosting** puede aprender la importancia relativa de cada señal y sus interacciones no lineales (p. ej., cómo la combinación de un dígito 'caliente' en una posición y una transición de Markov específica afecta la probabilidad final).[14] Esto permite que el sistema aprenda patrones más complejos que una simple suma ponderada, al tiempo que se mantiene la transparencia al poder inspeccionar la importancia de cadafeature para el modelo.[17]
- **Módulo de Evaluación y Monitoreo (Load):** Este servicio, programado para ejecutarse al día siguiente (T+1), cierra el ciclo de feedback. Compara los candidatos generados con los resultados reales del sorteo.[1] Sus métricas de éxito incluyen:
  Hit exacto, Aciertos parciales y el Lift vs azar (comparando el rendimiento del modelo con una línea base aleatoria).[1] Con base en estas métricas, el módulo sugiere ajustes automáticos de los pesos de las señales, con un límite de cambio (
  rate limiting) para garantizar la estabilidad.[1]

- **Módulo de Reporting:** Este servicio genera reportes diarios y semanales en formato HTML.[1] La información se presenta de manera transparente, incluyendo los candidatos propuestos, los resultados reales, las métricas de acierto y una sección de "Qué funcionó / Qué falló" que justifica los cambios propuestos en los pesos de las señales.[1] Es crucial que cada reporte incluya un
disclaimer sobre el juego responsable, reforzando la postura ética del sistema.[1]

3.3 El Cerebro del Sistema: Un Enfoque de MLOps

La propuesta de arquitectura se alinea con las mejores prácticas de MLOps (Machine Learning Operations), un conjunto de principios para la gestión del ciclo de vida de los modelos de aprendizaje automático en producción.[19] Esto aborda directamente los desafíos de reproducibilidad y trazabilidad.

- **Versionado de Datos y Modelos:** La reproducibilidad del sistema depende de tres componentes: el código, los datos y los parámetros del modelo. La versión del código se gestiona con Git. Para los datos y los metadatos del modelo (signals_config), se propone el uso de **Data Version Control (DVC)**.[21] DVC, al integrarse con Git, permite versionar archivos grandes de datos (
datasets) y artefactos del modelo. Cada vez que se realiza un experimento o se actualiza una señal, se crea una nueva versión en DVC, que se enlaza al commit de Git.[21] Esto garantiza que cualquier
run del pasado pueda ser recuperado y replicado con la misma versión de código y datos, lo que es esencial para la auditoría y la explicación del rendimiento.[21]
- **Trazabilidad y Data Lineage:** La trazabilidad es el rastro que sigue un dato desde su origen hasta su destino. El concepto de data lineage proporciona un registro auditable de cómo los datos se transforman en el pipeline.[22] En 'Radar Premios', esto significa registrar la
semilla usada para la generación, el snapshot de las señales y sus pesos (signals_snapshot) y las decisiones tomadas en cada etapa.[1] Esto es vital para depurar errores, validar la integridad de los datos y responder a la pregunta de por qué un modelo funcionó de una manera específica en una fecha determinada.[23]
- **Monitoreo del Rendimiento y Detección de Drift:** Un modelo en producción necesita ser monitoreado continuamente para asegurar su rendimiento.[19] Se propone la implementación de un
dashboard de monitoreo de KPIs.[25] Además de las métricas de acierto, es crucial monitorear la
data drift, o la deriva en la distribución de las features de entrada, y la prediction drift, o la deriva en la distribución de los resultados predichos.[27] Un cambio significativo en la

distribución de los números ganadores, por ejemplo, podría indicar que las señales actuales ya no son relevantes y que el modelo necesita ser re-entrenado o ajustado.[27] Se pueden usar pruebas estadísticas como el test de Chi-cuadrado o el test de Kolmogorov-Smirnov (
KS test) para detectar estas desviaciones de manera automática y generar alertas proactivas.[1]

4.0 Plan de Acción: Roadmap de Implementación por Fases

El plan de acción se presenta como un roadmap estructurado y secuencial, garantizando una implementación metódica y un avance incremental hacia la visión final del sistema.

Tabla 4: Roadmap Detallado por Fases

Fase	Hito Clave	Tecnologías Relevantes	Objetivo Principal
Fase 1: Modernización del Entorno (MVP Operacional)	Containerización de scripts existentes con Docker. Migración de schtasks a Docker Compose. Implementación inicial de un DAG en Apache Airflow.	Docker, Docker Compose, Python, Apache Airflow	Un sistema operacional y reproducible en cualquier máquina.
Fase 2: Fortalecimiento y Rigor Científico	Implementación de matrices de Markov posicionales. Configuración de signals_config y signals_history en	DVC, Pandas, Numpy	Un sistema con un fundamento matemático avanzado y auditable.

	la base de datos. Integración de DVC para versionar datos y modelos.		
Fase 3: Optimización y Monitoreo Avanzado	Implementación de Montecarlo baseline para medir Lift. Creación de dashboard de monitoreo de KPIs. Automatización del ajuste de pesos de señales. Detección de data y prediction drift.	Librerías de monitoreo (e.g., EvidentlyAI), modelos de ML (e.g., Scikit-learn).	Un sistema autónomo, auto-ajustable y con alertas proactivas.

5.0 Métricas de Éxito, Ética y Conclusiones

5.1 KPIs de Rendimiento del Pipeline

Para medir el éxito del sistema, se propone un conjunto de indicadores clave de rendimiento (KPIs) que cubren tanto los aspectos técnicos como el impacto del modelo.[25] Estos KPIs se integrarán en un

dashboard para monitoreo continuo.

Tabla 3: KPIs de Rendimiento del Pipeline (Módulo de Monitoreo)

KPI	Descripción	Valor Actual (Fase	Objetivo

		Inicial)	
Disponibilidad del Pipeline	Porcentaje de runs de generación de candidatos completados exitosamente en un día.	N/A	>99%
Latencia de Proceso	Tiempo promedio de ejecución del pipeline completo (ingesta, generación, evaluación).	N/A	< 1 hora
Trazabilidad	Porcentaje de runs con semilla y snapshot de señales guardados.	N/A	100%
Lift vs. Azar	Tasa de acierto del modelo sobre la tasa de acierto de una simulación aleatoria.	N/A	>1
Recall@K	Tasa de acierto del premio dentro del Top-K de candidatos.	N/A	Mejorar gradualmente
Estabilidad de Señales	Correlación entre el score de una señal y el acierto real del sorteo.	N/A	> 0.5

5.2 Transparencia y el Compromiso Ético

La naturaleza del sistema requiere un compromiso inquebrantable con la transparencia y la ética. La promesa de 'Radar Premios' no es garantizar un premio, sino proporcionar una herramienta de análisis estadístico. Cada reporte, tanto diario como semanal, debe incluir una declaración clara que indique que "RadarPremios es un sistema de análisis estadístico. No garantiza premios. Juega responsablemente".[1] Se debe incluir un número de teléfono o local de ayuda para la ludopatía en el pie de página.[1] Esta medida no solo cumple con un requisito de responsabilidad, sino que también refuerza la posición del sistema como una herramienta científica, distanciándose de la superstición y el pensamiento irracional asociados a los juegos de azar.

5.3 Conclusión: Posicionando a 'Radar Premios' para el Futuro

La arquitectura y el plan de acción aquí presentados marcan una transición fundamental para 'Radar Premios': de un proyecto de scripts dependiente de una máquina individual a una plataforma de MLOps robusta, escalable y auditable. Esta evolución no solo resuelve los desafíos operativos existentes, sino que también establece una base sólida para el crecimiento futuro. La integración de tecnologías como Docker y Apache Airflow asegura que el sistema pueda escalar para manejar nuevos juegos de lotería o un mayor volumen de datos sin comprometer su resiliencia. El uso de DVC, junto con el monitoreo avanzado de rendimiento y deriva, garantiza que el sistema siga siendo un artefacto científico, donde cada decisión y resultado es trazable y reproducible.

En última instancia, 'Radar Premios' se posiciona como un ejercicio de ciencia aplicada. Su verdadero valor reside en su capacidad para modelar la incertidumbre, validar hipótesis y, en el proceso, promover una visión más racional y transparente del juego. La estrategia propuesta asegura que el sistema siga operando en la intersección del rigor matemático y la responsabilidad ética, creando un producto que no solo busca modelar el azar, sino también educar sobre su naturaleza.

Obras citadas

1. Radar Premios — Arquitectura Lógica Y Plan De Integración De Señales.pdf
2. Windows Task schedular on Docker - General, fecha de acceso: septiembre 1, 2025, https://forums.docker.com/t/windows-task-schedular-on-docker/129050
3. ¿Qué es la arquitectura de microservicios? - Google Cloud, fecha de acceso: septiembre 1, 2025, https://cloud.google.com/learn/what-is-microservices-architecture?hl=es
4. ¿Qué es la arquitectura de microservicios? - Google Cloud, fecha de acceso: septiembre 1, 2025,

https://cloud.google.com/learn/what-is-microservices-architecture?hl=es-419
5. ETL/ELT - Apache Airflow, fecha de acceso: septiembre 1, 2025, https://airflow.apache.org/use-cases/etl_analytics/
6. Scheduling Data Pipelines with Apache Airflow: A Beginner's Guide - DASCA, fecha de acceso: septiembre 1, 2025, https://www.dasca.org/world-of-data-science/article/scheduling-data-pipelines-with-apache-airflow-a-beginners-guide
7. Markov Orden Superior And Datos Históricos - FasterCapital, fecha de acceso: septiembre 1, 2025, https://fastercapital.com/es/palabra-clave/markov-orden-superior-and-datos-hist%C3%B3ricos.html
8. Cadenas de Markov en Python con ejemplos de modelos | DataCamp, fecha de acceso: septiembre 1, 2025, https://www.datacamp.com/es/tutorial/markov-chains-python-tutorial
9. ¿Qué son y para qué sirven los microservicios? - Red Hat, fecha de acceso: septiembre 1, 2025, https://www.redhat.com/es/topics/microservices
10. Docker y Herramientas de Orquestación | Decimo Sexta Edición - IoT en la actualidad, fecha de acceso: septiembre 1, 2025, https://revistaecys.github.io/16Edicion/08_rcutz.html
11. DAGify: Enterprise schedule migration accelerator for Airflow - YouTube, fecha de acceso: septiembre 1, 2025, https://www.youtube.com/watch?v=xxP4UknkmBo
12. Las 20 principales herramientas de ingestión de datos en 2025: La guía definitiva, fecha de acceso: septiembre 1, 2025, https://www.datacamp.com/es/blog/data-ingestion-tools
13. ¿Qué es la ingestión de datos? - insightsoftware, fecha de acceso: septiembre 1, 2025, https://insightsoftware.com/es/encyclopedia/what-is-data-ingestion/
14. 7 Mínimos cuadrados no lineales | Machine Learning: Teoría y Práctica - Bookdown, fecha de acceso: septiembre 1, 2025, https://bookdown.org/victor_morales/TecnicasML/m%C3%ADnimos-cuadrados-no-lineales.html
15. Redes neuronales: Funciones de activación | Machine Learning - Google for Developers, fecha de acceso: septiembre 1, 2025, https://developers.google.com/machine-learning/crash-course/neural-networks/activation-functions?hl=es-419
16. Gradient boosting - Wikipedia, fecha de acceso: septiembre 1, 2025, https://en.wikipedia.org/wiki/Gradient_boosting
17. Líneas de producto para automatizar la combinación de modelos de Machine Learning, fecha de acceso: septiembre 1, 2025, https://ingenieriadesoftware.es/lineas-de-producto-para-automatizar-la-combinacion-de-modelos-de-machine-learning/
18. Aprendizaje automático: Qué es y por qué importa - SAS, fecha de acceso: septiembre 1, 2025, https://www.sas.com/es_es/insights/analytics/machine-learning.html
19. Introducción al monitoreo del rendimiento de los modelos (MLOps), fecha de acceso: septiembre 1, 2025,

https://docs.newrelic.com/es/docs/mlops/get-started/intro-mlops/
20. ¿Qué es MLOps? - IBM, fecha de acceso: septiembre 1, 2025, https://www.ibm.com/mx-es/think/topics/mlops
21. Guía del versionado de datos para MLOps usando DVC, fecha de acceso: septiembre 1, 2025, https://www.enmilocalfunciona.io/versionado-de-datos-para-mlops-con-dvc/
22. About data lineage | Dataplex Universal Catalog - Google Cloud, fecha de acceso: septiembre 1, 2025, https://cloud.google.com/dataplex/docs/about-data-lineage
23. What Is Data Lineage? | IBM, fecha de acceso: septiembre 1, 2025, https://www.ibm.com/think/topics/data-lineage
24. Evaluar el rendimiento del modelo a lo largo del tiempo | Qlik Cloud Ayuda, fecha de acceso: septiembre 1, 2025, https://help.qlik.com/es-ES/cloud-services/Subsystems/Hub/Content/Sense_Hub/AutoML/models-over-time.htm
25. AI & ML KPIs - Top 12 AI & Machine Learning KPIs - Drive AI Success with AssessTEAM, fecha de acceso: septiembre 1, 2025, https://www.assessteam.com/ai-ml-kpis/
26. How to Monitor Manufacturing KPIs Automatically with Machine Learning - Acerta Analytics, fecha de acceso: septiembre 1, 2025, https://acerta.ai/articles/how-to-monitor-manufacturing-kpis-automatically-with-machine-learning/
27. What is data drift in ML, and how to detect and handle it - Evidently AI, fecha de acceso: septiembre 1, 2025, https://www.evidentlyai.com/ml-in-production/data-drift
28. Detecting data drift using Amazon SageMaker | AWS Architecture Blog, fecha de acceso: septiembre 1, 2025, https://aws.amazon.com/blogs/architecture/detecting-data-drift-using-amazon-sagemaker/