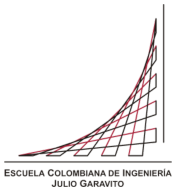

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

Repo-Aura

Proyecto de curso PTIA

Grupo 02

Autor:
Andrés Felipe Chavarro Plazas

Fecha de entrega: 28 de octubre del 2025

Declaración firmada

"Declaro que he escrito este trabajo de investigación por mí mismo, y que no he utilizado otras fuentes o recursos que los indicados para su preparación. Declaro que he indicado claramente todas las citas directas e indirectas, y que este documento no se ha presentado en otro lugar para fines de examen o publicación"

Nombre del estudiante & Firma:

Andrés Felipe Chavarro Plazas

Fecha: 16 de septiempre del 2025

Índice de contenido

1. Introducción

En el ámbito de la ingeniería de software actual, la gestión de proyectos se enfrenta a una brecha crítica: la dependencia de métricas de resultados que actúan como indicadores de éxito dentro de este, que a menudo revelan disfunciones cuando su impacto en el cronograma y la calidad es ya ineludible. Este proyecto se postula como una respuesta a dicha brecha, proponiendo un sistema proactivo que aproveche la riqueza de datos generados durante el ciclo de vida del desarrollo para ofrecer una visión predictiva de la salud del mismo. El enfoque se centra en la resolución de tres problemáticas interconectadas y fundamentales:

Primero, la **alta tasa de fracaso en proyectos de software**, impulsada por la acumulación de deuda técnica y la erosión de la colaboración dentro de los equipos, factores que las métricas de rendimiento convencionales no logran capturar de manera efectiva. Segundo, la dificultad inherente a la detección temprana de riesgos centrados en el capital humano, como la sobrecarga de personal y la degradación del ambiente compartido de trabajo, los cuales son precursores directos de la ineficiencia y la rotación. Tercero, la **subjetividad en la evaluación de la salud integral de un proyecto**, que actualmente recae en la intuición, en lugar de en un análisis cuantitativo, continuo y basado en evidencia.

La naturaleza de estos desafíos excede las capacidades de la computación tradicional, posicionando a este proyecto como una aplicación idónea para la materia **Principios de Técnicas de Inteligencia Artificial (PTIA)**. La solución exige una arquitectura de Inteligencia Artificial capaz de modelar sistemas socio-técnicos complejos, integrando tanto heurísticas expertas como patrones de datos temporales. Por un lado, se requiere la implementación de un sistema basado en conocimiento para codificar la heurística experta en la identificación de patrones de riesgo conocidos. Por otro, es indispensable el uso del aprendizaje automático, específicamente de redes neuronales recurrentes, para discernir patrones complejos y tendencias en secuencias de datos heterogéneos como código, texto y metadatos. Así, este trabajo no solo aborda una problemática de alta relevancia, sino que también funciona como una síntesis práctica de los paradigmas fundamentales de la inteligencia artificial.

2. Trabajos relacionados

El análisis de repositorios de software para extraer conocimiento accionable es un campo consolidado, conocido como **Minería de Repositorios de Software** (MSR). Las herramientas y trabajos existentes han abordado la predicción de la salud de un proyecto desde diferentes ángulos, enfocándose en la calidad del código estático o en métricas de productividad. Este proyecto, **RepoAura**, se posiciona en la intersección de estos enfoques, proponiendo un análisis holístico que integra la dimensión técnica con la humana y temporal.

2.1. Predicción de Defectos y Deuda Técnica

- **Herramienta - SonarQube:** Es la plataforma líder para el análisis estático de código. Su enfoque es identificar bugs y code smells basándose en reglas predefinidas sobre el estado actual del código. Su principal limitación es que, si bien rastrean la evolución de la deuda técnica, actúan como un historial o una "fotografía" del estado actual, no como un modelo predictivo de tendencias futuras complejas. RepoAura se diferencia al usar la evolución temporal con secuencias para predecir problemas futuros....
- **Trabajo Académico - "Predicting Defects for Eclipse":** Este trabajo seminal demostró que las métricas históricas como el "churn" de código son fuertes predictores de defectos, usando modelos estadísticos. RepoAura se diferencia al extender esta premisa integrando factores humanos y usando modelos más especializados con RNNs para capturar la dinámica temporal de forma más sofisticada.

2.2. Análisis de Factores Humanos y de Colaboración

- **Herramienta - Pluralsight Flow:** Plataforma de "Developer Intelligence" que mide la a productividad del equipo. Su enfoque es la eficiencia y la velocidad de desarrollo. RepoAura se diferencia al centrarse en el riesgo y la salud social y técnica. No se pregunta "¿cuán rápido vamos?", sino "¿estamos en riesgo de fallar?".
- **Trabajo Académico - "The Impact of Sentiment on Software Development":** Esta línea de investigación utiliza Procesamiento de lenguaje natural para correlacionar el sentimiento negativo en las comunicaciones con la introducción de bugs. RepoAura se diferencia al no solo mostrar una correlación, sino al integrar el sentimiento como una característica clave dentro de un modelo predictivo holístico que también considera métricas de código y actividad.

3. Criterios y Problema Seleccionado

La elección del problema se fundamentó en un conjunto de criterios diseñados para garantizar su viabilidad, relevancia y rigor académico, asegurando que la solución sea técnicamente desafiante y aplicable a la industria del software.

3.1. Criterios de Selección

- **Relevancia Industrial:** Se priorizó un problema cuya solución tuviera un impacto tangible en la industria, abordando la necesidad real de mitigar los altos índices de fracaso en proyectos. El objetivo es superar las métricas reactivas tradicionales mediante la detección proactiva de riesgos que preceden a desviaciones en costo y calidad.
- **Alineación Académica y Complejidad Técnica:** El problema posee una complejidad inherente que justifica un enfoque de IA híbrido. Requiere la aplicación integrada de sistemas basados en reglas para codificar heurísticas expertas y de aprendizaje automático para identificar patrones complejos en los datos que los métodos tradicionales no pueden capturar, justificando su desarrollo en el marco del curso.
- **Disponibilidad de Datos:** Un criterio pragmático fue la existencia de conjuntos de datos públicos y ricos, disponibles en repositorios de código abierto. Esta accesibilidad es un pilar indispensable para el entrenamiento y la validación de cualquier modelo de aprendizaje supervisado, eliminando una barrera crítica del proyecto. Sin embargo, dado que los modelos de aprendizaje supervisado requieren etiquetas que no existen públicamente, un criterio adicional fue la viabilidad de generar etiquetas proxy, como la mortalidad del proyecto o la densidad de defectos post-release, y a partir de los metadatos históricos obtenidos, permite el entrenamiento del modelo
- **Potencial de Innovación:** Se buscó un enfoque que, si bien se basa en trabajos existentes, ofrece una perspectiva holística novedosa. La innovación radica en combinar el análisis de las dimensiones técnica, colaborativa y humana en un único modelo, tratando el proyecto como un sistema socio-técnico integrado.

3.2. Problema Seleccionado

Considerando los criterios expuestos, **el problema seleccionado es la detección proactiva del riesgo y la predicción de la salud integral en proyectos de desarrollo de software**. Este desafío se abordará mediante la construcción de un modelo de aprendizaje automático multifactorial, entrenado para identificar las señales débiles y los patrones sutiles que preceden a los fallos. El objetivo final es desarrollar un sistema de alerta temprana que permita pasar de una gestión de proyectos reactiva a una gobernanza proactiva y guiada por datos.

4. Justificación del problema

El gran problema que este proyecto busca resolver es la **naturaleza reactiva y a menudo tardía de la gestión de riesgos en el desarrollo de software**. Los métodos convencionales (como diagramas de Gantt o burndown charts) son fundamentalmente indicadores rezagados. Incluso las métricas modernas de CI/CD como las métricas DORA, aunque son excelentes para medir el rendimiento del proceso, ejemplo: la velocidad de entrega y la estabilidad, fallan en capturar la salud interna del sistema socio-técnico que lo produce. Informan de problemas —como una caída en la velocidad del equipo o un aumento en la tasa de errores— solo después de que la enfermedad subyacente ya ha comenzado a impactar el rendimiento, sea la deuda técnica oculta, la fricción colaborativa, entre otros. En esencia, la gestión actual detecta los síntomas, no la causa raíz.

Es de vital importancia resolver este problema por dos razones estratégicas. Primero, en un entorno económico donde la eficiencia y la velocidad de comercialización son críticas, **la capacidad de anticipar fallos en lugar de reaccionar a ellos representa una ventaja competitiva decisiva**. Permitiría a las organizaciones mitigar riesgos proactivamente, optimizar la asignación de recursos y, en última instancia, reducir los costos asociados con el retrabajo, los retrasos y los fracasos de proyectos. Segundo, y quizás más importante, este enfoque pone el foco en el **capital humano**, que es el activo más valioso en el desarrollo de software. Al identificar señales de sobrecarga, frustración o fricción en la colaboración, se pueden tomar acciones para mejorar el bienestar del equipo, prevenir el agotamiento (burnout) y fomentar un entorno de trabajo más sostenible y productivo. Resolver este problema, por lo tanto, no solo mejora el producto, sino que protege y potencia a las personas que lo construyen.

5. Alcance y Objetivos

5.1. Alcance: El Problema Efectivo a Resolver

Considerando las limitaciones de un proyecto académico de un mes, el alcance se limita a **dos fases críticas**. En primer lugar, un proceso de definición y validación de una etiqueta proxy que represente un "riesgo de salud" de forma verdaderamente predictiva. En segundo lugar, la construcción y validación de un prototipo de modelo de aprendizaje automático (RNN) capaz de predecir esta etiqueta basándose en los mencionados datos históricos socio-técnicos.

5.2. Justificación de la Simplificación

Esta simplificación es adecuada y necesaria por dos motivos alineados con los objetivos del curso y las restricciones del proyecto:

- **Enfoque en la Complejidad de IA**: El alcance se centra en el núcleo de la complejidad de PTIA, que es demostrar la viabilidad de la predicción y, crucialmente, la generación de una etiqueta predictiva viable. Se excluyen tareas de alta complejidad de ingeniería de software como plataformas de streaming en tiempo real o UIs empresariales, que no aportan valor a la validación de la hipótesis de IA.
- **Viabilidad en el Tiempo**: Desarrollar una plataforma de producción es un esfuerzo de meses. Un prototipo que demuestre la capacidad predictiva sobre un conjunto de datos estático y una etiqueta proxy bien definida es un objetivo tangible y académicamente riguroso que puede completarse en el tiempo asignado.

5.3. Objetivos

Objetivo Cualitativo

El objetivo cualitativo es demostrar la viabilidad de un enfoque de IA híbrido y socio-técnico para la gobernanza proactiva de proyectos. Se busca validar la hipótesis de que la combinación de **métricas técnicas y humanas** en un modelo temporal (RNN) ofrece una capacidad predictiva superior a la de los modelos estáticos tradicionales, que analizan estos factores de forma aislada.

Objetivo Cuantitativo

El objetivo cuantitativo es desarrollar un modelo predictivo usando **Redes Neuronales Recurrentes**, que demuestre una mejora de rendimiento estadísticamente significativa (medida por F1-Score o AUC-ROC) en comparación con un modelo baseline. Este ultimo, se basará únicamente en métricas técnicas estáticas, mientras que el modelo propuesto (RNN) integrará la dimensión socio-técnica y temporal.

6. Análisis y Diseño Metodológico

6.1. Análisis de la solucion

Estrategia General de Solución

La estrategia general es un enfoque de **Inteligencia Artificial Híbrida** aplicado sobre el campo de la Minería de Repositorios de Software (MSR). La justificación de esta estrategia se basa en la naturaleza dual del problema. Ningún enfoque único es suficiente:

- **Sistema Basado en Conocimiento (Reglas)**: Es necesario para codificar heurísticas expertas y deterministas sobre el estado actual del código. Por ejemplo, un alto nivel de complejidad ciclómática es un riesgo técnico.
- **Aprendizaje Automático (ML)**: Es indispensable para modelar la dinámica temporal y los patrones emergentes de los factores humanos y de colaboración. Por ejemplo, un declive sutil pero sostenido en el sentimiento de los issues predice una futura caída en la productividad.

El prototipo se centrará en el componente de Aprendizaje Automático, ya que representa el núcleo de la innovación y la mayor complejidad de IA.

Modelo Conceptual del Agente (PEAS)

Para formalizar la solución como un sistema de IA, definimos el agente *RepoAura* usando el modelo PEAS (Performance, Environment, Actuators, Sensors):

- **Performance:** La medida de éxito es la mejora significativa del F1-Score del modelo socio-técnico propuesto en comparación con un modelo baseline.
- **Environment:** El ecosistema socio-técnico de los repositorios de software públicos, presentes en platadormas como GitHub o GitLab. Es un entorno parcialmente observable, pues no vemos las conversaciones offline, estocástico impredecible y secuencial.
- **Actuators:** Para este prototipo, el actuador es la generación de una salida de clasificación de riesgo. Ejemplo, RIESGO-BAJO: 0.1, RIESGO-ALTO: 0.9. En una implementación futura, podrían ser alertas proactivas a los líderes de equipo.
- **Sensors:** Las APIs de los repositorios para ingerir commits, pull requests, churn de código, y las herramientas de análisis de texto para extraer características de sentimiento de issues y comentarios.

Concentración de la Inteligencia

La inteligencia de la solución se divide en dos componentes:

- **Conocimiento**: El conocimiento se materializa en la base de datos de entrenamiento. Esta no es solo una colección de datos, sino un conjunto estructurado de secuencias temporales de características,

sean series de tiempo del churn, sentimiento, frecuencia de commits, entre otros, que se encuentran mapeadas a las etiquetas proxy de riesgo que se definieron en el alcance.

- **Razonamiento**: El razonamiento se concentra en el modelo de Redes Neuronales Recurrentes (RNN). Este motor de inferencia realiza un razonamiento temporal complejo, capaz de identificar las dependencias a largo plazo entre variables socio-técnicas heterogéneas para predecir un estado de riesgo futuro.

Herramientas Seleccionadas y Justificación

- **Lenguaje y Ecosistema: Python 3.12:** Es el estándar de facto para la ciencia de datos y el machine learning. Su ecosistema es indispensable para la fase de MSR y análisis, con tecnologías como Pandas para manipulación de datos y Matplotlib o Seaborn para visualización.
- **Framework de ML: TensorFlow con Keras.** Para un proyecto académico de este estilo, la alta abstracción de Keras es crucial para prototipar, entrenar y evaluar rápidamente arquitecturas de redes neuronales complejas (como LSTMs o GRUs) sin perder tiempo en implementaciones de bajo nivel.
- **Baseline y Preprocesamiento: Scikit-learn.** Es esencial para crear el modelo baseline con Regresión Logística o Random Forest, contra el cual se medirá el éxito del objetivo cuantitativo.

6.2. Diseño de la Solución

Arquitectura de la Solución (Prototipo)

La arquitectura del prototipo sigue un diseño de Pipeline de Procesamiento de Datos por Lotes (Batch), enfocado en la validación del modelo, no en la inferencia en tiempo real. Los componentes y sus relaciones son:

1. **Módulo de Ingesta y Extracción (MSR)**:
   - **Responsabilidad**: Conectarse a las APIs de repositorios públicos en plataformas para extraer datos crudos como commits, issues, PRs y comentarios de un conjunto seleccionado de proyectos.
   - **Salida**: Datos crudos (JSON, texto).
2. **Módulo de Ingeniería de Características y Etiquetado (ETL)**:
   - **Responsabilidad**: Transformar los datos crudos en un conjunto de datos tabular y secuencial. Esto incluye calcular métricas (ej. churn-rate, pr-velocity) y ejecutar análisis de sentimiento sobre el texto.
   - **Responsabilidad crítica**: Generar la etiqueta proxy basada en el estado futuro del proyecto
   - **Salida**: Vectores de características y etiquetas.
3. **Módulo de Entrenamiento (IA)**:

- **Responsabilidad**: Ingerir los datos procesados. Entrenar y evaluar dos modelos en paralelo; el Baseline y el Modelo Propuesto (RNN).
- **Salida**: Modelos entrenados (.h5 o .pt).

4. **Módulo de Evaluación y Reporte**:
   - **Responsabilidad**: Comparar el rendimiento (F1-Score, AUC-ROC) de ambos modelos usando un conjunto de prueba (Test Set).
   - **Salida**: El resultado del objetivo cuantitativo, por ejemplo, indicar que el modelo RNN superó al baseline en un 15 %.

Componente Principal: Modelo Predictivo Socio-Técnico (RNN)

El componente principal y el núcleo de la inteligencia de la solución es el Modelo Predictivo basado en Redes Neuronales Recurrentes, implementando específicamente una arquitectura GRU (Gated Recurrent Unit) como modelo primario.

- **Descripción**: A diferencia de los modelos baseline que ven los datos como una "fotografía" estática, el modelo GRU está diseñado para procesar secuencias de datos.
- **Justificación de la Arquitectura**: La hipótesis del proyecto es que el riesgo no depende del estado actual, sino de la tendencia y la interacción de factores a lo largo del tiempo. Se selecciona GRU sobre LSTM como punto de partida por tres razones:
  - Su eficiencia computacional permite una iteración más rápida, crucial para un proyecto de un mes, pues tiene menos parametros.
  - Su simplicidad reduce el riesgo de sobreajuste en un conjunto de datos MSR limitado.
  - Su rendimiento es empíricamente comparable al de LSTM para la mayoría de tareas.
- **Ejemplo de Razonamiento**: Un modelo estático no puede diferenciar entre un proyecto con 100 bugs que está mejorando (resolviendo 20 por día) y uno que está empeorando (agregando 10 por día). Una GRU, al tener memoria, puede aprender estas dependencias temporales. Es capaz de modelar cómo un declive en el sentimiento hace 30 días, combinado con un aumento en la complejidad del código hace 15 días, predice de manera conjunta una caída en la velocidad de resolución de PRs mañana. Esta capacidad de razonamiento temporal es la clave para la predicción proactiva.

7. Referencias bibliográficas

Begel, A., Nagappan, N., Poile, C., & Layman, L. (2008). Coordination in Large-Scale Software Development: Helpful and Unhelpful Behaviors. En *Proceedings of the 2008 ACM/IEEE 30th International Conference on Software Engineering (ICSE '08)* (pp. 191–200). Association for Computing Machinery. https://doi.org/10.1145/1368088.1368115

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. En *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1179

Hassan, A. E. (2008). The Road Ahead for Mining Software Repositories. En *Proceedings of the 2008 Frontiers of Software Maintenance (FOSM '08)* (pp. 48–57). IEEE Computer Society. https://doi.org/10.1109/FOSM.20

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Lin, B., & Serebrenik, A. (2018). Sentiment Analysis for Software Engineering: How Far Can We Go?. En *Proceedings of the 15th International Conference on Mining Software Repositories (MSR '18)* (pp. 80–84). Association for Computing Machinery. https://doi.org/10.1145/3196398.3196434

Zimmermann, T., & Weissgerber, P. (2004). Preprocessing CVS Data for Fine-Grained Analysis. En *Proceedings of the 1st International Workshop on Mining Software Repositories (MSR '04)* (pp. 2–6). Association for Computing Machinery. https://doi.org/10.1145/1083142.1083145

Zimmermann, T., Premraj, R., & Zeller, A. (2007). Predicting defects for eclipse. En *Proceedings of the Third International Workshop on Predictor Models in Software Engineering (PROMISE '07)* (p. 9). IEEE Computer Society. https://doi.org/10.1109/PROMISE.2007.10