ARTÍCULO 1

Ferreira, I., Fernandes, E., Cedrim, D., Uchôa, A., Bibiano, A. C., Garcia, A., ... & de Mello, R. (2018, May). The buggy side of code refactoring: Understanding the relationship between refactorings and bugs. In *Proceedings of the 40th international conference on software engineering: companion proceeedings* (pp. 406-407).

https://dl.acm.org/doi/pdf/10.1145/3183440.3195030?casa\_token=-vCNvOH1zEQAAAAA:1z8e4zFhW2cw3yCj57qQrrp2uRQ6H3a-nF5GK46En4YAQCCwK8VXckHdspH6lvuX2TMk1zjg4ZGgMg

Este estudio examina cómo los errores y la refactorización de código se relacionan entre sí. Las operaciones de refactorización en cinco proyectos de Java fueron examinadas por los investigadores, donde se descubrió que los elementos de código que han sido refactorizados tienen más probabilidades de contener errores introducidos poco después de la refactorización. Además, observaron que la refactorización del código junto con otros cambios en el código (floss refactoring) tiende a presentar errores antes que la refactorización del código aislada (root-canal refactoring). Según este estudio, aunque el refactoring se considera bueno, en realidad puede causar nuevos errores, especialmente cuando se realiza junto con otros cambios en el código. Los investigadores sugieren que los desarrolladores deben ser conscientes de estos riesgos y tomar medidas para evitarlos.

ARTÍCULO II

DROSOS, G. P., SOTIROPOULOS, T., SPINELLIS, D., & MITROPOULOS, D. Bloat beneath Python’s Scales: A Fine-Grained Inter-Project Dependency Analysis.

https://dimitro.gr/assets/papers/DSSM24.pdf

El estudio analiza el impacto del código no utilizado proveniente de dependencias en proyectos de Python. Este código, conocido como "código de dependencia inflado", es común en el ecosistema Python y puede causar problemas de eficiencia y seguridad. Descubrieron que los proyectos Python suelen tener más de diez dependencias infladas, lo que puede introducir vulnerabilidades de seguridad que, aunque actualmente no son explotables, podrían serlo en el futuro. Los cambios generalizados en las bases de código, como la eliminación de funciones, son la principal causa del código inflado. Sin embargo, si los desarrolladores comprenden las causas de las dependencias infladas, suelen estar dispuestos a eliminarlas. Los investigadores utilizaron gráficos de dependencia de grano fino (FPDG) para evaluar el código inflado. Según el estudio, abordar el código inflado es esencial para mejorar la eficiencia y la seguridad del código Python.

ARTÍCULO III

Gardey, J. C. (2024). Methods and tools for the development and evaluation of refactorings to improve the user experience in web applications. *Journal of Computer Science and Technology*, *24*(1), e06-e06.

<https://journal.info.unlp.edu.ar/JCST/article/download/3166/1887>

La investigación se enfoca en mejorar la evaluación de la experiencia del usuario (UX) en aplicaciones web. Aunque existen métodos de evaluación automática de UX, carecen de la capacidad de validar cambios de diseño con los usuarios. Para abordar esto, los autores proponen UX-Painter, un herramienta para los diseñadores de UX que pueden crear versiones alternativas de aplicaciones web mediante la refactorización (cambios menores de código) sin necesidad de escribir el código ellos mismos. Esto permite a los diseñadores visualizar y evaluar fácilmente los cambios en el diseño. También introducen el concepto de "esfuerzo de interacción" para medir el esfuerzo del usuario al interactuar con elementos web. Más tarde, esta métrica se utiliza en UX-Analyzer, otro instrumento que calcula y compara el esfuerzo de interacción de diferentes diseños de aplicaciones web. Finalmente, los autores muestran métodos para crear código para aplicar estas mejoras de experiencia de usuario en la aplicación real. Este artículo sugiere un conjunto de herramientas y técnicas para que los diseñadores UX evalúen y mejoren la experiencia del usuario en aplicaciones web, haciéndolo más eficiente y efectivo en metodologías ágiles.