CODE REVIEWS

¿QUÉ SON LAS REVISIONES DE CÓDIGO?

Nacen con las metodologías ágiles, más en concreto con SCRUM. Se trata de un proceso mediante el cual uno o varios miembros de un equipo revisan la implementación realizada por otro miembro del mismo. El objetivo debe ser determinar qué se está haciendo bien para seguir haciéndolo y qué se puede mejorar; nunca tratar de buscar culpables.

Ayuda a mejorar la comunicación del equipo, la calidad del código del proyecto o la calidad del código de los miembros entre otras cosas. Con las revisiones de código se pueden evitar problemas como son la *propiedad de código* o el *Developer Guilt.*

¿QUÉ SE MIRA EN LAS REVISIONES DE CÓDIGO?

Arquitectura/Diseño

- **Principio de única responsabilidad:** Una clase debe tener una sola responsabilidad. Lo mismo debería pasar con los métodos.
- **Principio abierto/cerrado:** Que los módulos estén abiertos a ser cambiados o extendidos en funcionalidad, pero cerrados a cambios del código fuente.
- **Duplicación de código:** Regla de los 3 strikes, si hay que copiar más de una vez un código será mejor refactorizar.
- **Ofensas visuales:** Mirando el código y entornando los ojos, ¿se ven formas o patrones conocidos que interesan o que se pretenden evitar?
- **Errores potenciales y manejo de errores:** Si hay posibles errores que no se hayan tenido en cuenta. Y los que sí, ¿se están manejando correctamente?
- **Eficiencia:** Que los algoritmos se estén utilizando de la manera más eficiente.

Estilo

- **Nombres de métodos:** Deben ser comprensibles y acordes a lo que hacen.
- **Nombres de variables:** Que sean sencillos y con sentido.
- **Longitud de las funciones:** Deberían tener menos de 20 líneas idealmente, pero nunca más de 50.
- **Longitud de las clases:** No deberían pasar de 300 líneas, lo ideal serían 100.
- **Docstrings:** Para comentar o explicar métodos complejos.
- **Código comentado:** Sería mejor evitarlo.
- **Número de argumentos en los métodos:** Tratar de evitar que haya más de 3.
- **Legibilidad:** Que sea fácil de leer.

Testing

- **Cobertura de tests:** Toda nueva funcionalidad debería tener sus tests.
- **Testing en la medida adecuada:** Se recomienda 95% de tests unitarios y 5% de tests de integración.
- **Número de mocks:** No debería haber más de 3 mocks por test.
- **Cumplir los requisitos:** Repasar los requisitos de la historia de usuario y comprobar que se cumplen.

Existen diferentes varientes de las revisiones de código, como pueden ser *The Email Thread, Pair programming, Over-the-shoulder* o *Tool-assisted.*

Natalia Fernández Riego UO277516
Alberto Fernández Azcoaga UO179637

BUENAS PRÁCTICAS

Mejoras prácticas para los autores de los cambios del código

- **Leer los cambios, agrupar relacionados, tests**

Para ahorrar tiempo a los revisores, mientras se prepara un cambio para una revisión, los autores deben ser conscientes y leer el cambio detenidamente antes de enviarlo para que se revise.

Además, también es una buena idea agrupar los cambios que están relacionados.

Los autores también deberían invertir tiempo en testear sus cambios.

- **Revisar antes política de revisión**

Los autores de cambios de código también deben considerar cuándo omitir una revisión al consultar la política de revisión de código de su organización (si existe).

- **Seleccionar los revisores**

Una vez el código ya ha sido preparado para una revisión, los autores deben seleccionar sus revisores. En particular, necesitan decidir cuantos revisores se necesitan, consultando la política de su organización si fuera necesario.

- **Notificar las correcciones**

Los autores también deben expresar gratitud a los revisores y considerar su feedback de una manera respetuosa.

Mejores prácticas para los revisores de código

- **Dedicarle tiempo**

Los revisores deben reservar tiempo para revisarlo, empleando el tiempo que se considere necesario para entender el código de cada revisión.

- **Feedback**

Para ayudar a los autores del código, también es muy importante proporcionar un feedback lo más rápido posible para que los autores puedan acordarse de sus cambios.

- **Checklist, canales de comunicación, opinión constructiva**

También se suele considerar el uso de una checklist de la revisión, hecha “a medida” para cada proyecto.

Mejores prácticas a considerar por los organizadores

- **Establecer una política de review**

Para maximizar el valor de las revisiones de código, una organización debería considerar establecer una política de code review. Esta política debería ayudar a construir una cultura de la revisión positiva que establezca las bases de un feedback constructivo.

- **Evitar impactos negativos**

Se recomienda recompensar a los ingenieros que dedican esfuerzo y tiempo en revisar el código de sus compañeros, penalizar a los ingenieros que no lo hacen (normalmente por una buena razón) puede llevar a una manipulación del sistema.

- **Usar las herramientas adecuadas**

También es importante asegurarse que se usan las herramientas adecuadas y que coinciden con la cultura de la revisión deseada y del proceso definido (si lo hay).

Las nuevas herramientas para apoyar las actividades de revisión de código están emergiendo todo el tiempo y es posible (y recomendable) que una organización quiera estar al tanto de estos desarrollos.

- **Ofrecer formación**

Para abordar los desafíos relacionados con conocer el proceso esperado o con cómo usar las herramientas deseadas, una organización tiene que asegurarse de que haya suficiente formación. Esta puede ser a través de tutorías con sus compañeros o formándose en internet.

Referencias:

- Breve explicación de Code Review escrita por varios mimebros de Microsoft: https://www.michaelagreiler.com/wp-content/uploads/2019/03/Code-Reviewing-in-the-Trenches-Understanding-Challenges-Best-Practices-and-Tool-Needs.pdf
- https://kinsta.com/es/blog/herramientas-de-revision-de-codigo/
- https://smartbear.com/learn/code-review/what-is-code-review/
- https://slashmobility.com/blog/2017/01/code-review-todo-lo-que-debes-saber/