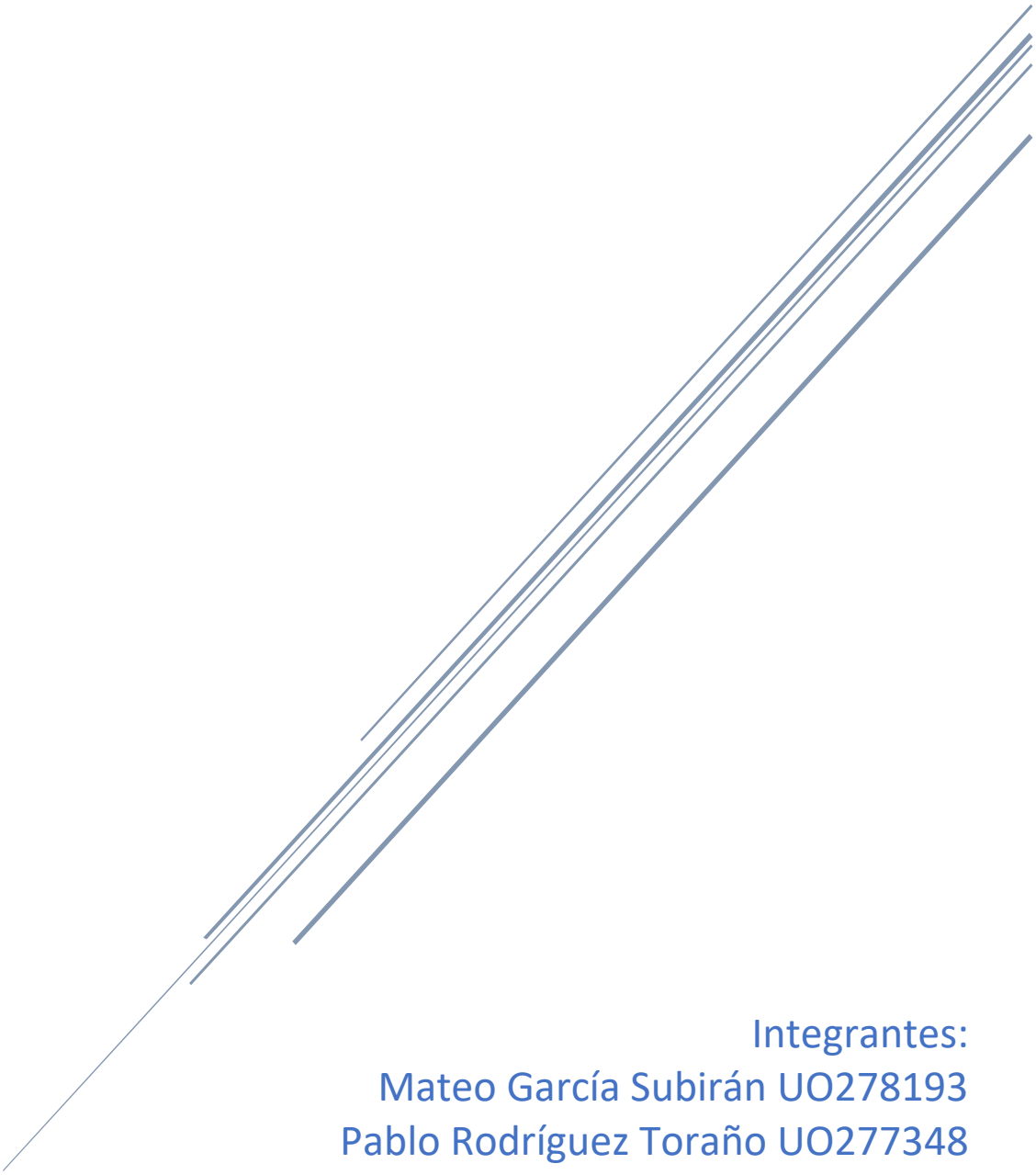
CODE REVIEW

Arquitectura del Software



Integrantes:
Mateo García Subirán UO278193
Pablo Rodríguez Toraño UO277348
Damir Abdrafikov UO277306
Valentín Dumitru UO277867

1. INTRODUCCION

El code review es el acto consciente y sistemático junto con el programador de comprobar el código de este en busca de errores que aparecen frecuentemente y buscar una solución para agilizar y acelerar el proceso de desarrollo software. Es una práctica ampliamente adoptada y realizada por empresas en sus proyectos, ya sean o no de código abierto.

El code review ha sido ampliamente estudiado con muchos estudios abalando su eficacia en el desarrollo de código, siendo una de las que mejores resultados dan.

El mayor problema del code review es que pese a su extensión, el code review es muy disperso y está mal extendido. Esto se debe a que existen diferentes estrategias en función del contexto y esto hace difícil su unificación.

2. CODE REVIEW STUDY

Para entender el proceso del code review, hay que ver el punto de vista de los revisores del código. Generalmente hacen un análisis retrospectivo del código a análisis, por ejemplo, mirando los commits de github. Otra opción es hacer encuestas o entrevistas para revelar los posibles desafíos que se encontraran durante la revisión.

Microsoft basa el code review en un ciclo de vida que incluye 6 pasos (Puntos dentro de Code review de Microsoft): Preparación, Selección, Notificacion, Feedback, Iteracion y Registro del cambio.

El ciclo varía en función del equipo que haga el code review, haciendo primero un paso o priorizando unos antes que otros. Además, usan la herramienta de code review CodeFlow que ayuda a seguir unos pasos durante la revisión de código.

Además de obtener ayuda de una aplicación existen otros enfoques del code review como por ejemplo el email thread, el pair programming o el over-the-shoulder.

Algo que tienen en común todos los enfoques, ya sea el de Microsoft u otro son los problemas que uno se encuentra mientras hace una revisión.

3. CODE REVIEW CHALLENGES

3.1 RETOS DE LOS AUTORES DE CODIGO (RESPONSABLE A CORREGIR)

3.1.1 RECIBIR FEEDBACK A TIMEPO.

A veces resulta complicado recibirla a tiempo, ocurre a menudo que el que se encarga de hacer la revisión tarde en exceso. Entorpeciendo el horario y la planificación del programador.

3.1.2 RECIBIR FEEDBACK UTIL Y RELEVANTE.

Ocurre a menudo que el encargado de revisar se fija en exceso en detalles insignificantes cuando debería buscar fijarse en grandes problemas.

3.1.3 ENCONTRAR REVISORES APROPIADOS Y QUE ESTEN DISPUESTOS.

Un problema externo también preocupante es la falta buenos revisores y disponibles.

3.1.4 DOCUMENTAR LOS CAMBIOS PARA LA REVISION.

Resulta que una gran cantidad de personas no documentan los cambios realizados en su código a la hora de mandar a revisar.

3.1.5 EL RECHAZO PUEDE SER DESAGRADABLE.

Es desagradable que rechacen tu código sin buenas razones, resulta mucho más amigable que den buenas razones.

3.1.6 LA REVISION RESULTA EFIMERA.

A veces es difícil recordar los problemas a la hora de corregirlos, especialmente cuando las revisiones se realizan mediante videollamadas o en persona. Es mejor usar aplicaciones dedicadas a ello, que registran las revisiones.

3.1.7 MANEJAR MULTIPLES CANALES DE COMUNICACION.

Proveniente del problema anterior, cuando las revisiones no se programan y organizan bien, acaba resultando difícil y estresante manejar todos los canales en los que se ha hecho la review.

3.1.8 UTILIZAR HERRAMIENTAS REALENTIZA EL PROCESO

A su vez proveniente del problema anterior, utilizar herramientas ralentiza la velocidad en la que se programa.

3.2 RETOS DE LOS REVISORES (RESPONSABLES DE REVISAR EL CODIGO)

1. Revisiones muy extensas
2. Encontrar tiempo para revisar
3. Entender el propósito, la motivación y el como del código.
   (Muchos códigos son una gran masa de texto incomprensible que resulta difícil de aportar valor a la hora de revisar.)
4. Encontrar documentación relevante sobre los cambios
   Este problema lo comparten las dos partes, una gran cantidad de personas no documentan los cambios necesarios en su código a la hora de mandar a revisar.
5. Entender la historia de los comentarios
6. Falta de entrenamiento del proceso de revisión
7. Manejar múltiples canales de comunicación.
8. La actividad de revisión no se valora lo suficiente.

4.BEST PRACTICIES

4.1 BEST PRACTICIES FOR CODE CHANGE AUTHORS

El autor deberá pasar una herramienta de revisión a su código para ahorrar tiempo al revisor, ya que estas detectan fácilmente errores pequeños. Una vez utilizada el autor debería crear alguna prueba para probar el correcto funcionamiento.

Los autores posteriormente seleccionaran a los revisores si las políticas de empresa lo permiten, la empresa concreta tratada recomienda que sean 2 personas las encargadas.

Una vez realizada la revisión los autores y revisores se mantendrán en contacto para aceptar y tratar los cabios propuestos. Estos deben estar documentados y justificados antes de ser confirmados y corregidos en la versión final.

4.2 BEST PRACTICIES FOR CODE REVIEWERS

El revisor debe revisar frecuentemente el código, pero revisando menos cambios a la vez, debe usar los mejores canales de comunicación para hablar con el autor. Se recomienda el cara a cara para discutir lo más complejo y usar canales que dejen una traza de los cambios para aquellos más delicados. Una vez realizado, el feedback se dará de la forma más respetuosa y constructiva, para la final revisión del autor.

4.3 BEST PRACTICIES FOR ORGANIZATIONS TO CONSIDER

Como una organización prepara el escenario para la revisión de código y la forma con la cual lo soporta y valúa las revisiones es un parte critica del éxito de las revisiones.

Una organización debería establecer una política de revisiones.

También es importante que se usen las herramientas apropiadas y que el proceso este bien definido. Deberían ser ligeras e integradas con otras herramientas, sobre todo herramientas de comunicación.

Las organizaciones también deberían tener en cuenta las nuevas herramientas y mantenerse al tanto de sus desarrollos.

Cualquier problema que ralentice la revisión debería ser resuelto de inmediato.

5. TRADEOFFS TO CONSIDER WHEN APLAYING BEST PRACTICIES

Algunas prácticas de revisión tienen conflictos entre sí. Por ello hay inevitablemente tenemos que elegir entre algunas de ellas para compensar.

Si hay falta de tiempo es posible que sea necesario hacer que las revisiones sean menos rigurosas para poder hacerlas más rápido.

Políticas rígidas como requerir autentificación en 2 pasos o verificar un set de pruebas unitarias pueden hacer que se tarde más en realizar un commit.

Políticas muy relajadas o poco comprensibles pueden hacer que el valor de las revisiones sea menor.

La única forma de controlar estos sacrificios es tener en cuenta que estos existen y cuales son, y buscar los que mejor se adapten a la forma de trabajo de tu equipo

6. TIPS

1. Hacer preguntas durante la revisión del código.
2. Aclara que es tu perspectiva.
3. No uses sarcasmo ni seas condescendiente.
4. Usa emojis en los comentarios
5. Explica tus razones de porque pediste el cambio