A continuación, se listan las reglas de Clean Code **mínimo** en Masivian:

1. Todo el código y comentarios están 100% en inglés, nada de español a excepción estricta de mensajes que se vayan a mostrar a usuarios finales en interfaz gráfica.
2. No dejar espacios en blanco entre líneas de código a excepción de una línea antes del return.
3. Los nombres de las funciones deben ser autodescriptivos, no deberían requerir un comentario que explique su función
4. Funciones con única responsabilidad, los principales indicadores de que no es de única responsabilidad son:
   - funciones de más de 25 líneas de código.
   - loops o condicionales anidados más de 1 vez.
   - secciones de código dentro de la función que requieran comentarios para ser comprendidas
5. Las variables de configuración deben ser leídas de variables de entorno y las variables de acceso (contraseñas, claves, nombres de usuario) deberían ser secrets de AWS
6. Tener un archivo por clase, es decir no poner más de una CLASS o ENUM en un solo archivo
7. Cero código comentado, el historial se lleva en Git, no en comentarios.
8. Las fechas se deben manejar, enviar y guardar en formato ISO 8601:
   - 2020-04-21T13:09:24-05:00 seria fecha y hora en UTC-5
   - 2020-04-21T18:09:24Z seria la misma fecha y hora, pero en UTC-0
9. Usar named paramaters al llamar métodos si el lenguaje lo permite (¿en que nos ayuda? R:/no depender del orden de entrega de los parámetros y poder hacer mejor uso de parámetros opcionales)
10. Evitar el uso de ORMs
11. Pasar Objetos y no Propiedades como parámetros, en la medida de lo posible
    ej: bien: CamapaingRepository.save(Campaign)
    mal: CamapaingRepository.save(id, status, createdAt, deliveriesArray)
12. Los Logs deben ir a ClowdWatch y el nivel mínimo de log debe ser configurable desde variable de entorno.

Notas:

- ¿Porque no tener espacios en blanco?
  R:/ Este post lo explica:
  https://www.yegor256.com/2014/11/03/empty-line-code-smell.html

- ¿Por qué no tener cometarios para los métodos y funciones?
  R:/ este post lo explica:
  https://blog.usejournal.com/stop-writing-code-comments-28fef5272752

- ¿Por qué no usar ORMs?
  R:/ En pocas palabras, porque suelen tener muchas ineficiencias y en operaciones de alto performance o complejidad traen más problemas de los que tratan de resolver. Igualmente este post lo profundiza:
  https://www.yegor256.com/2014/12/01/orm-offensive-anti-pattern.html

- ¿Como saber si estamos haciendo objetos con sentido?
  R:/ los objetos tienen comportamientos autónomos y no imperativos (por ejemplo un objeto de Ordenes autónomo tiene un método de creación que realiza validaciones automáticamente sin que quien lo llame tenga que validar por su cuenta si la orden es válida)
  https://www.yegor256.com/2014/11/20/seven-virtues-of-good-object.html