Manifiesto de Codificación: Guía para un Desarrollo Ágil y Eficaz

En el dinámico mundo del desarrollo de software, la calidad y la eficiencia son pilares fundamentales para el éxito de cualquier proyecto. Es por ello que el Manifiesto de Codificación surge como una guía invaluable, estableciendo principios y buenas prácticas que permiten a los desarrolladores crear aplicaciones robustas, escalables y mantenibles.

Principios

1. La simplicidad como brújula:

Evite la complejidad innecesaria en su código. Opte por soluciones simples, directas y fáciles de entender. Recuerde: “Lo que se puede expresar de manera sencilla, no se debe complicar”.

2. Priorice la legibilidad:

Escriba código claro y conciso que se explique por sí mismo. Utilice nombres de variables descriptivos, comentarios relevantes y una estructura organizada. Un código legible facilita la comprensión, el mantenimiento y la colaboración.

3. Pruebas unitarias: la base de la confianza:

Implemente pruebas unitarias exhaustivas para cada componente de su código. Estas pruebas automatizadas le brindarán la seguridad de que cada módulo funciona correctamente, aislando errores y previniendo regresiones.

4. CI/CD: automatización a su servicio:

Adopte prácticas de Integración Continua (CI) y Entrega Continua (CD) para automatizar el proceso de construcción, pruebas y despliegue de su software. Esto agiliza el desarrollo, reduce errores y garantiza una entrega constante de valor al usuario.

5. Documentación: la memoria del código:

Documente su código de manera completa y precisa. Explique el propósito de cada módulo, las funciones de cada clase y las decisiones de diseño tomadas. Una documentación clara facilita la comprensión del código para otros desarrolladores y futuros colaboradores.

6. Despliegue controlado:

Implemente estrategias de despliegue controladas que minimicen el riesgo de errores en producción. Utilice entornos de preproducción para realizar pruebas exhaustivas antes de liberar el software a los usuarios finales.

7. La refactorización: un arte para mejorar:

No tema refactorizar su código para mejorar su estructura, legibilidad y mantenibilidad. La refactorización continua le permite mantener un código limpio y eficiente a lo largo del tiempo.

8. El código limpio es un código feliz:

Establezca estándares de codificación claros y consistentes dentro de su equipo. Utilice herramientas de análisis de código estático para identificar y corregir posibles errores de sintaxis, estilo y seguridad.

9. La colaboración: la clave del éxito:

Fomente la colaboración y el intercambio de conocimientos entre los miembros del equipo. Realice revisiones de código periódicas para identificar áreas de mejora y aprender unos de otros.

10. La retroalimentación: un regalo invaluable:

Busque y reciba retroalimentación constante sobre su código. Escuche atentamente las sugerencias de sus compañeros, usuarios y clientes. La retroalimentación le permite identificar oportunidades de mejora y adaptar su código a las necesidades reales.

Políticas

- Todo microservicio del negocio debe generarse por medio de arquetipo.
- Todo microservicio debe entregarse a la fábrica registrado en el repositorio de la organización.
- Todo microservicio debe disponer de profiles de configuración para los ambientes definidos (default, dev, qa*, pre*, prod*)
- Todo microservicio debe disponer de profiles de configuración CI/CD para los ambientes definidos (dev, qa*, pre*, prod*)
- Las entidades que hacen persistencia deben mapear todo la clase que representan.
- Las consultas sql que devuelven data compleja (más de una propiedad) deben tener su entidad y utilizar las anotaciones @Immutable, @SubSelect.
- Las entidades solo se pueden utilizar en clases @Component y no pueden ser utilizadas en otras clases superiores con anotaciones @RestController, @Service.
- Las entidades solo pueden pertenecer a un solo microservicio el cual se encargará de la

responsabilidad de persistencia.
- Las entidades solo pueden exponerse por medio de DTOs (Data-Transfer-Object) el cual pueden devolver información parcial/total según se requiera.
- Las entidades del negocio no deben utilizar funcionalidades de un motor de base de datos especifico, solo debe tener código SQL estandar.
- Las clases servicios deben utilizar la anotación @Service y solo pueden orquestar los métodos de la solucion.
- Las clases controladoras deben utilizar la anotación @RestController y solo pueden exponer endpoints.

Recuerde: El Manifiesto de Codificación no es una lista rígida de reglas, sino una guía flexible que se adapta a las necesidades de cada proyecto. Adopte estos principios y adáptelos a su contexto para crear software de alta calidad que resista el paso del tiempo.

←Regresar

From:
http://wiki.adacsc.co/ - **Wiki**

Permanent link:
http://wiki.adacsc.co/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:manifestcode&rev=1717763685

Last update: **2024/06/07 12:34**