

Disminuyendo la carga cognitiva de
nuestro código a través del Clean Code

.NET Conf 2023

Comunidad
Sabados Tech



¿Quién soy?

- Leonardo Micheloni
- Programador +20 años
- Microsoft MVP 9 años
- Arquitecto y tech lead en Tokiota Madrid



@leomicheloni

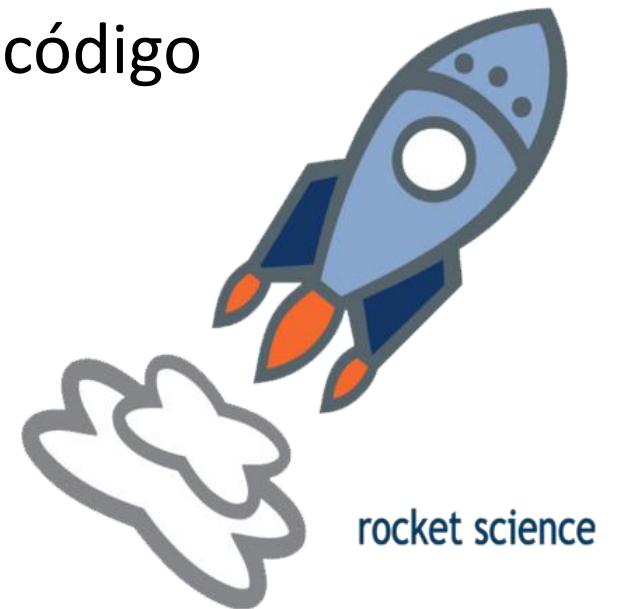


@leomicheloni



¿Por qué son importantes estos conceptos?

- Un framework / lenguaje / library no es más que una herramienta
- Se habla de ser bueno usando x pero no de ser buen programador
- Es cada vez más común mezclar tecnologías y herramientas
- Las aplicaciones evolucionan cada vez más rápido
- No siempre somos nosotros quienes comenzamos el código





Types of Headaches

Migraine



Hypertension



Stress



El variable se llama
strNombre



"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler





DRY

- **Don't repeat yourself principle**
- Es similar a la normalización de bases de datos
- Copy + Paste oriented programming
- Hay más líneas de código para mantener

極度乾燥(しなさい)
Superdry.

KISS

- **Keep it simple stupid**
- Hay un momento en nuestra evolución que nos gusta ser complicados
- La simplicidad es belleza



YAGNI

- **You ain't gonna need it**
- Voy a hacer tal cosa por las dudas
- No se extremistas



Simple, direct, prose

Clean code is simple
and direct

Clean code reads like
well-written prose



Grady Booch

Code Smells

- Olores de código
- Indican que el código tiene potenciales problemas
 - Para ser mantenido
 - Para escalar
 - Para evolucionar
- Es propenso a errores



Clean code: Naming

- Deben ser claros en su intención
- Deben indicar responsabilidad única
- No deben tener prefijos o subfijos
- Las variables booleanas deben responder preguntas positivas



Clean code: Naming

- Smells
 - AND OR IF
 - El código tiene “side effects”
 - Tipos de datos en el nombre
 - Booleanos que no responden una pregunta
 - Variables o métodos que no son simétricos
 - Funciones con efectos secundarios no descriptos por su nombre



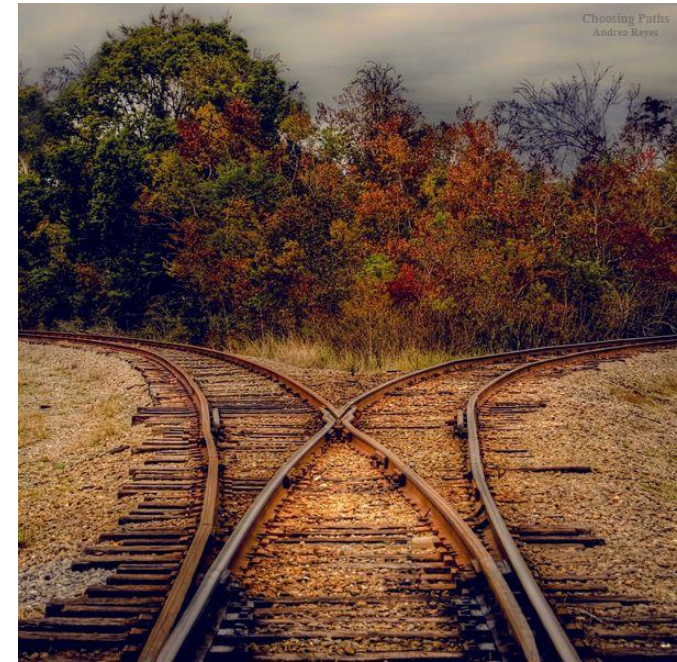
Clean Code: Naming

- Soluciones
 - Verbalizar con un amigo
 - O con un pato de goma
 - Refactorizar hasta que el nombre sea “limpio”



Clean Code: Condicionales

- Comparar explícitamente
- Usar condicionales positivos (evitar la doble negación)
- Asignar implícitamente
- Evitar “magic numbers”
- Evitar condicionales complejos



Clean Code: Condicionales

- Smells
 - Ser “antinegativo”
 - Magic numbers
 - Asignaciones de booleanos en condiciones
 - Comentarios sobre una condición
 - Condiciones difícil de comprender



Clean Code: Condicionales

- Soluciones
 - Siempre usar booleanos que respondan preguntas positivas
 - Poner magic numbers en variables
 - Crear funciones si la condición es una regla de negocio
 - Utilizar variables intermedias





Clean Code: Funciones

- Evitar Arrow Code
- Evitar funciones con demasiada responsabilidad
- Evitar funciones con efectos secundarios
- Evitar muchos argumentos
- Evitar funciones con flag arguments

Arrow code

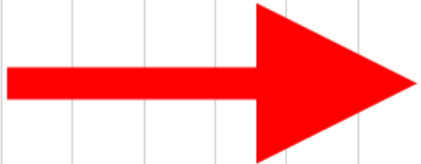
- Identificación excesiva
- Complejo de comprender
- Complejo de modificar

```
var result = "";

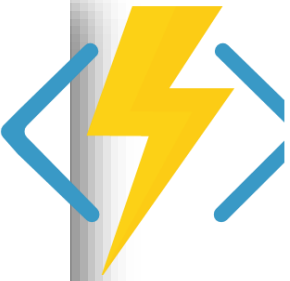
if(!user.isRegistered()){
  if(user.name.length < 3){
    result = "the name is too short";
  }else{
    if(user.password.length < 4){
      result = "the password is too short";
    }else{
      if(user.age < 18){
        result = "you must be over 18 in order to register my friend";
      }else{
        if(user.gender == ""){
          result = "A gender must be specified";
        }else{
          if(user.country == ""){
            result = "Please select a country from the list";
          }else{
            if(user.isValidEmail()){
              result = "Welcome man";
            }else{
              result = "The email is not valid";
            }
          }
        }
      }
    }
  }
}

else{
  result = "This user is already registered";
}

return result;
```



@leomicheloni





Funciones con efectos secundario

- El nombre no denota intención
- Hay miedo a cambiarlas
- Son peligrosas

```
function RegisterUser(userToRegister){  
  var r = registrationService.registerUser(userToRegister);  
  if(r == true){  
    mailing.sendConfirmation(userToRegister.email);  
    pushService.notify(userToRegister.email);  
    history.delete();  
  }else{  
    logger.error("Error in regsitration");  
  }  
}
```

Funciones con mucha responsabilidad

- Son difíciles de mantener
- Son difíciles de comprender
- Cambiar con mucha frecuencia



```
Receipt.prototype.calculateGrandTotal = function () {  
    var nSubTotal = 0,  
        nTax,  
        nTotal,  
        nLenTotals = this.aItemTotals.length,  
        nItemValue,  
        nDiscount,  
        nLenDiscounts = this.aDiscounts.length,  
        nDiscountValue;  
    for (nTotal = 0; nTotal < nLenTotals; nTotal++) {  
        nItemValue = this.aItemTotals[nTotal];  
        nSubTotal += nItemValue;  
    }  
    if (nLenDiscounts > 0) {  
        for (nDiscount = 0; nDiscount < nLenDiscounts; nDiscount++) {  
            nDiscountValue = this.aDiscounts[nDiscount];  
            nSubTotal -= nDiscountValue;  
        }  
    }  
    nTax = nSubTotal * 0.065;  
    nSubTotal += nTax;  
    return nSubTotal  
};
```



Clean code: Funciones

- Smells
 - Hay que hacer scroll para leerlas, vertical / horizontal
 - Tienen demasiados parámetros
 - Intención poco clara
 - Cambian con mucha frecuencia





Clean Code: Funciones

- Soluciones
 - Guard clauses
 - Fail fast
 - Extract method
 - Nuevas clases
 - Rename

Clean Code: Comentarios

- Pueden indicar que nuestro código no es claro
- Evitar comentarios obvios
- Evitar comentarios desactualizados



@leomicheloni



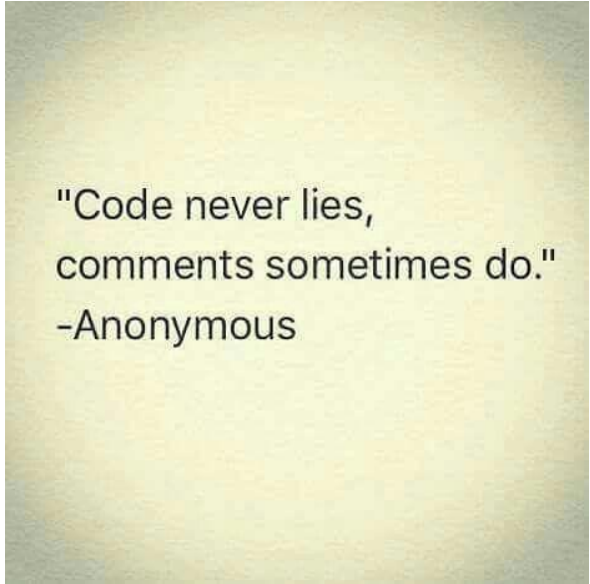
Clean Code: Comentarios

- Smells
 - Los comentarios son más grandes que el código
 - El comentario dice lo mismo que el nombre de la función
 - El comentario intenta explicar algo que el código debería
 - El comentario indica miedo



Clean Code: soluciones

- Al escribir un comentario pensar por qué lo hacemos
- Cambiar la función / variable para que indique su intención
- Borrar comentarios en condiciones y extraer la condición

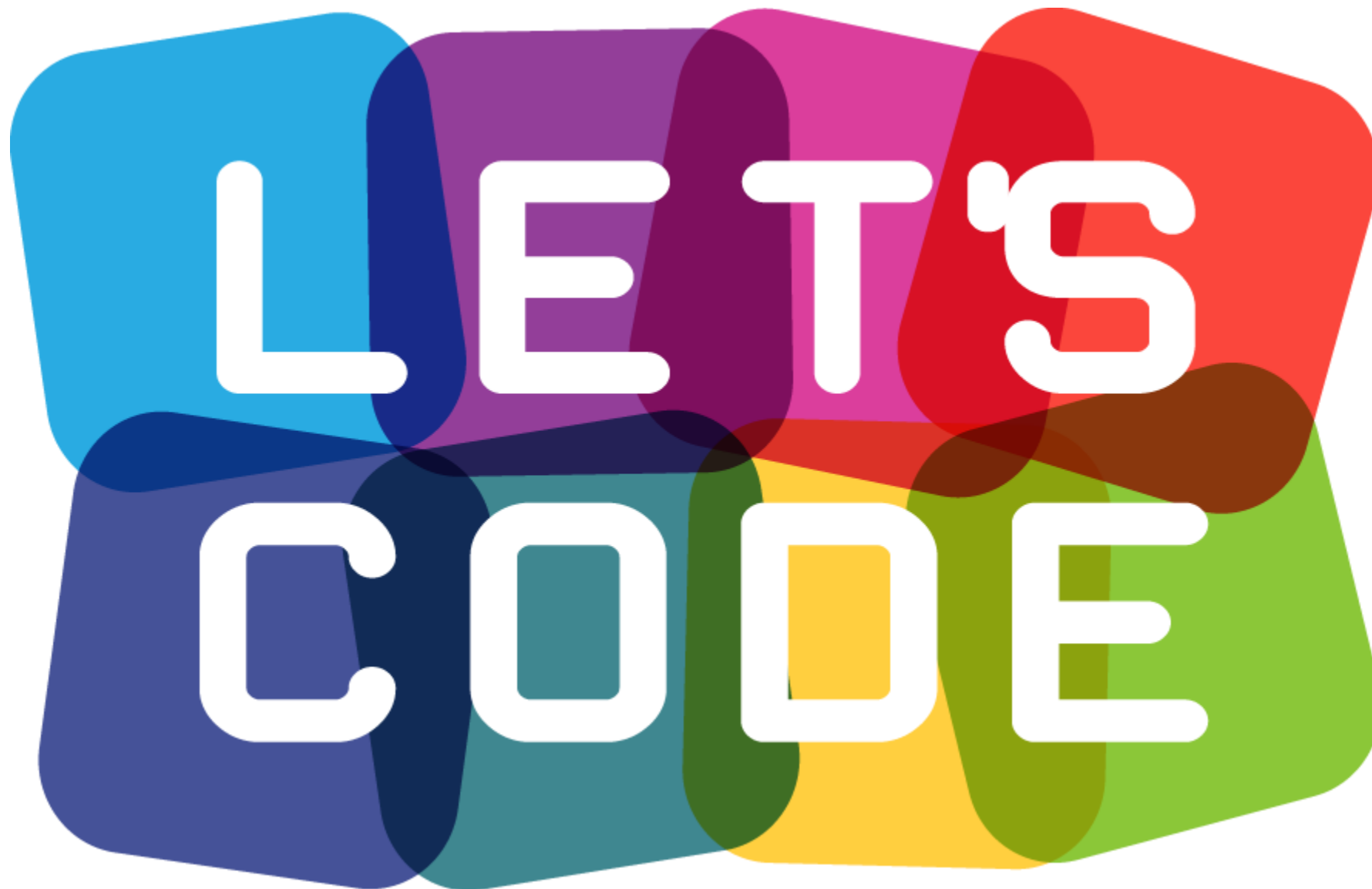


"Code never lies,
comments sometimes do."
-Anonymous

Refactor

- Cambiar el código sin modificar su comportamiento
- Hacerlo más legible
- Lo detectamos gracias a Code Smells, Patrones, DRY, etc.
- Aplicar algunos de los conceptos
 - KISS
 - DRY
 - SOLID
- Baby steps
- El mejor refactor es el que quita código





Mind set

- Mirar el código
- No odiar
- Respect
- Baby steps



Mantenimiento

- Refactoring
- Baby steps
- No aceptar ventanas rotas
- Code reviews
- Pair programming

“Always leave the code you're editing a little better than you found it.”

Robert C. Martin



@leomicheloni

Herramientas

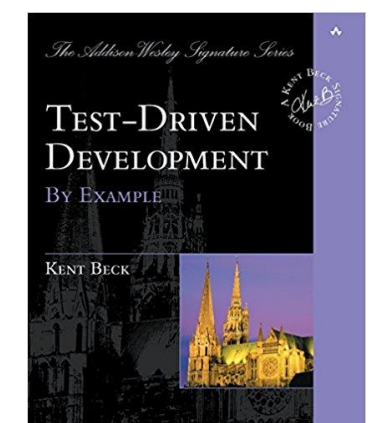
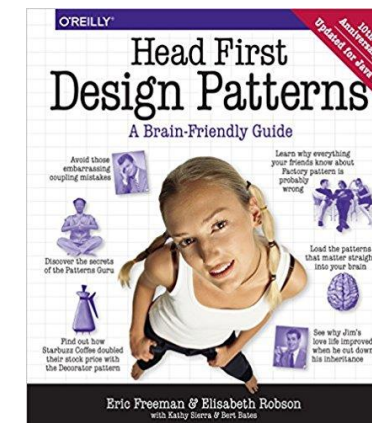
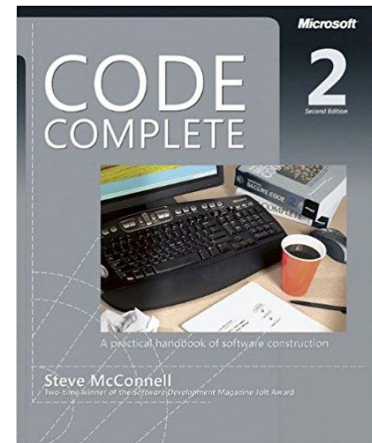
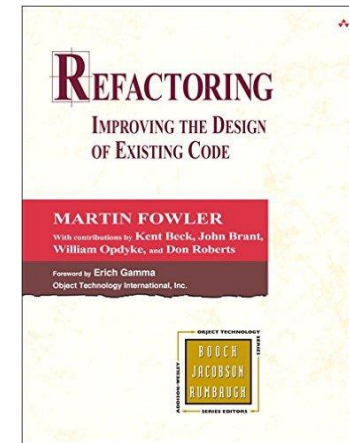
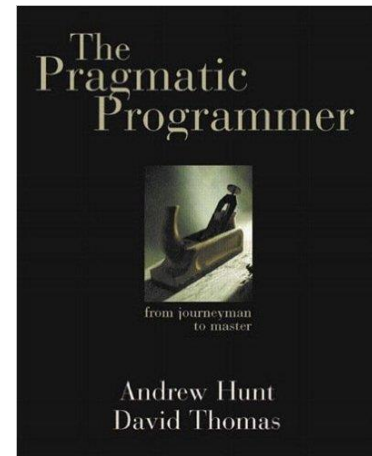
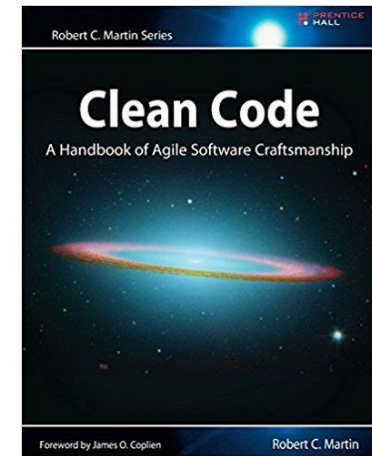
- Visual Studio
- Sonar Cube
- Code review
- Pair review

John Lennon (tal vez)

“Life is what happens to you while you’re busy trying to understand your own code”

Referencias

- <https://github.com/leomicheloni/sabados-tech-clean-code>
- <https://app.pluralsight.com/library/courses/writing-clean-code-humans>
- <https://www.amazon.es/Refactoring-Improving-Design-Existing-Technology/dp/0201485672/>
- <https://www.amazon.es/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882>
- <https://www.amazon.es/Code-Complete-Practical-Costruction-Professional/dp/0735619670/>
- <https://www.amazon.es/Pragmatic-Programmer-Journeyman-Master/dp/020161622X/>
- <https://www.amazon.es/Head-First-Design-Patterns-Freeman/dp/0596007124/>
- https://github.com/tcorral/Refactoring_Patterns



¿Preguntas?