Coding Bootcamp

Sprint 2

Temario

Temario

- ¿Cómo hemos organizado nuestro código hasta ahora?
- Atributos de calidad.
- ¿Qué lineamientos podemos seguir?
- Patrones de Diseño.
- MVC.

¿Cómo hemos organizado nuestro código hasta ahora?

¿Cómo hemos organizado nuestro código hasta ahora?

Como se ha mencionado anteriormente, Express es un framework minimalista y no-opinado.

Por lo tanto, Express no indica cómo organizar nuestro código. Por ejemplo, no indica qué archivos crear ni en qué carpetas colocarlos. El desarrollador tiene mucha libertad, lo cual conlleva a una gran responsabilidad.

Esto puede ser problemático en caso de programadores inexpertos ya que fácilmente pueden terminar con un código totalmente desorganizado, difícil de entender, difícil de mantener y tal vez hasta inseguro y poco performante.

Atributos de calidad

Atributos de calidad

Al hora de desarrollar software, se busca que el mismo cumpla ciertos atributos de calidad (también llamados características “no funcionales” del sistema).

Algunos de los más comunes son:

- Modificabilidad / Mantenibilidad.
- Rendimiento (Performance).
- Confiabilidad (Reliability).
- Escalabilidad.
- Disponibilidad.
- Usabilidad.
- Seguridad.
- Desplegabilidad (Deployability).

En general, se intenta que un software cumpla la mayor cantidad posible de atributos de calidad, aunque según el sistema, a veces se le suele dar más prioridad algunos atributos que a otros.

Incluso, a veces hay atributos que se contraponen. Por ejemplo, con el fin de que un sistema sea más seguro se podría perder usabilidad.

Un libro famoso al respecto es Software Architecture in Practice.

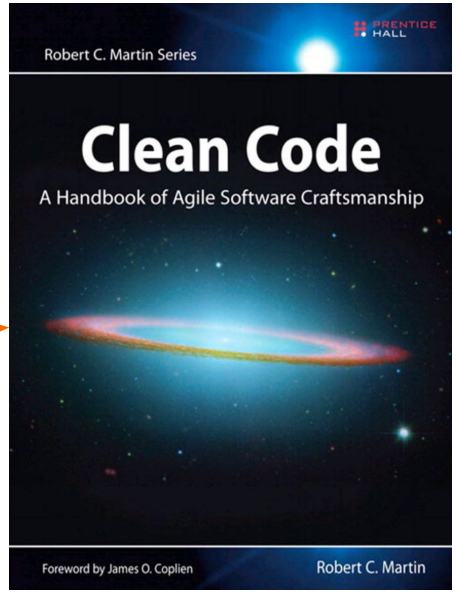
¿Qué lineamientos podemos seguir?

¿Qué lineamientos podemos seguir? (1/3)

Hay mucha bibliografía sobre cómo escribir código de calidad.

Por ejemplo, existen los llamados “principios” de desarrollo de software. Algunos conocidos son:

- DRY: Don't Repeat Yourself.
  ≈ DIE: Duplication is Evil.
- KISS: Keep it simple, Stupid!
- SOLID: Conjunto de 5 principios que buscan lograr un software más mantenible.
  Es un término acuñado por Robert C. Martin.
- GRASP: General Responsibility Assignment Software Patterns.



¿Qué lineamientos podemos seguir? (2/3)

Algunos lineamientos generales que pueden empezar a seguir son:

- El código simple siempre es mejor que el código complejo.
- Cuando detecten un problema, siempre busquen la causa raíz del mismo.
- El software debe ser fácilmente configurable. Evitar el “hard-codeo”.
- Sean consistentes. Si hacen algo de cierta manera, mantenerlo de esa forma a lo largo de todo el código.
- Usar nombres descriptivos y no ambiguos.
- Las funciones deben ser pequeñas, hacer una sola cosa, no tener efectos secundarios.

¿Qué lineamientos podemos seguir? (3/3)

- El código debería ser auto-descriptivo. No escriban comentarios redundantes u obvios.
- Funciones dependientes deberían “estar cerca” en el código.
- Funciones similares deberían “estar cerca” en el código.
- Los objetos/clases/módulos deberían ser pequeños y tener una única responsabilidad.
- Testeen su código.

Patrones de Diseño

Patrones de Diseño

Un patrón de diseño es una solución reusable a un problema común que se da en el desarrollo de software.

El término ganó popularidad en el año 1994, cuando se publicó el libro “Design Patterns: Elements of Reusable Object-Oriented Software” por “Gang of Four” (Gamma et al.), frecuentemente abreviado como “GoF”.

Los patrones no son soluciones “exactas”, son un esqueleto.

👉 Aquí puede ver un libro online sobre patrones, con foco en JavaScript.

MVC

MVC (1/3)

MVC es un **patrón de diseño arquitectónico** usado por muchos frameworks de desarrollo.

MVC propone dividir una aplicación en **3 grandes componentes** con responsabilidades bien definidas con el objetivo de fomentar el reuso de código y permitir el desarrollo en paralelo (se puede trabajar en simultáneo en cada componente).

Los componentes son: Modelos, Vistas y Controladores.

MVC (2/3)

Modelo: Es el componente central del patrón MVC. Expresa el comportamiento de la aplicación en términos del problema de negocio, de forma independiente a la interfaz (no importa si es una web, desktop app o mobile app). Gestiona los datos y reglas del negocio. En general interactúa con una base de datos.

Vista: Se encarga de presentar los datos del modelo. Puede ser una web en HTML, una mobile o desktop app o incluso la pantalla de un cajero automático.

Controlador: Es un intermediario. Recibe un request de un usuario a través de una URL. Se comunica con los modelos para obtener la información necesaria, la organiza y se la entrega a la vista.

MVC (3/3)

Resumen: