Decisiones de Diseño

ENTREGA 0

¿Cómo modelamos clientes y administradores?

- *Que extiendan de una clase usuario*
    - Comparten muchos campos
    - Ahorramos en getters y setters
    - Pero no comparten nada de lógica de negocios, no están relacionados
    - Al no compartir lógica, probablemente no habrá una función que necesite un “Usuario” (supuesta superclase del cliente y el administrador).
- *Dos clases individuales*
    - Reduce acoplamiento
    - Facilita mantenimiento
    - Aumenta cantidad de boilerplate (getters y setters)

¿Cómo manejamos las categorías?

- *Un string hardcodeado*
    - Poco extensible
    - Dificil de mantener
- *Polimorfismo*
    - Clases que solo tengan getter y setter
    - Lógica de la facturación es extraída al cliente.
    - Extensible
    - Dificil de mantener
    - Casi lo mismo que un string hardcodeado
- *Strategy*
    - Extensible
    - Mantenible
    - Modular
    - Muchísimo boilerplate
    - Clase abstracta o interface?
        - Prácticamente lo mismo, creemos que es mejor una clase abstracta porque comparten el método `facturacionPorMes()`, y en el futuro podrían compartir más lógica de algún otro método