# Patrones de diseño

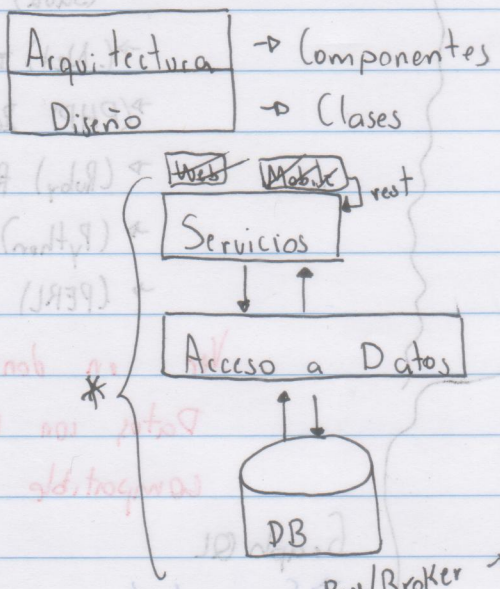
4-09-19

- Regularidad
- Discernible

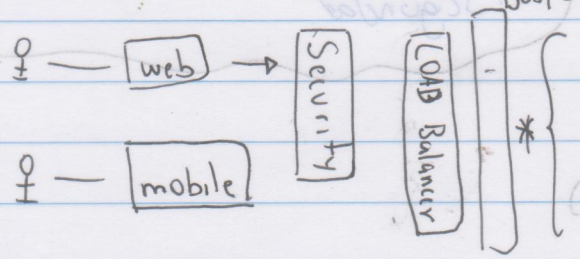
> Con la experiencia

- Patrón de diseño: es una buena práctica, solución probada
  - ⇒ Repositorio de conocimiento

- Patrones de arquitectura

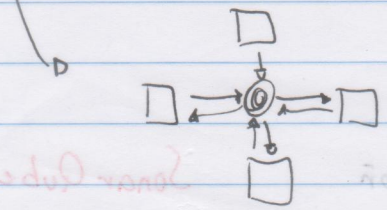



Building blocks




→ comunicación entre componentes (middle ware)

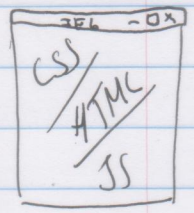
- MVC (Model - View - Controller)
- Publish / Suscribe — Midde ware
- SOA → Service Oriented Architecture



- Full-Stack Developer

Hace desde la gráfica hasta los datos

Front-End




- Vainilla JS/HTML/CSS
- BootStrap → CSS + HTML
- JQuery → Biblioteca HTML
- Angular
- VUE

JS → Ecma-Script 6

- React → FB

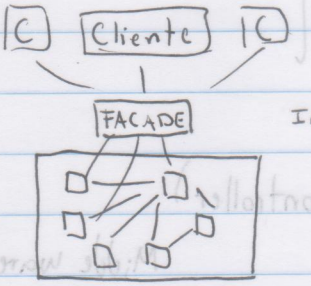
Back-end

- Apache HTTP Server (HTTPD) — WEB SERVER
- Rest-API => Application Server (Lenguaje)
  - → NODEJS (JS)
  - → (Java) Liberty, JBoss, Tomcat
  - → (.Net) IIS
  - → (PHP) Zend Server
  - → (Ruby) Rails
  - → (Python) Django
  - → (PERL)

Ver en donde se almacenan Datos con cuál es más compatible

GraphQL → Seguridad

→ Facade / Façade

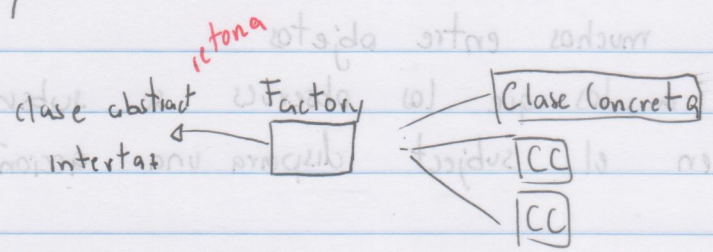



Interactua con clases en orden específico

* Típicamente es de terceros cuando es externo es por convención

Sonar Qube

- Factory



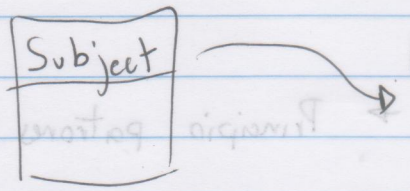

cliente no conoce clases concretas

Reutilizar código → Principio patrones diseño.

- Singleton
  - Tener un único punto de acceso a un recurso
  - Instanciación sale caro


```
class Singleton {
    private static Singleton instance = null;
    private Singleton() {
    }
                    ↗ synchronized (con hilos)
    public static Singleton getInstance() {
        se verifica si instancia es nula
        la crea, sino solo la retorna
    }
```

- Observer
  - Relación uno a muchos entre objetos
  - Hay subjects a los que los observes se subscriben
  - Un cambio en el subject dispara una acción a los observers.




Design Patterns Gang of Four