# ***ARQUITECTURAS SOFTWARE***

## ***ISMAEL VERDUGO CASTRO***

**AQUITECTURA DE DISEÑO INTERPRETER**

http://migranitodejava.blogspot.com/2011/06/interpreter.html

https://reactiveprogramming.io/blog/es/patrones-de-diseno/interpreter

**ARQUITECTURA EN PIZARRA**

http://nixonsistemas.blogspot.com/p/arquitectura-de-pizarra.html

https://aprendearquitecturasoftware.wordpress.com/2018/10/05/patron-pizarra-blackboard-grupo-8/

**ARQUITECTURA DIRIGIDA POR EVENTOS**

https://apiumhub.com/es/tech-blog-barcelona/arquitectura-dirigida-por-eventos/

https://www.lainnovacionnecesaria.com/arquitectura-dirigida-por-eventos/

## ***ÁLVARO MERINO MOLINA***

**ARQUITECTURA ORIENTADA A SERVICIOS**

https://www.ecured.cu/Arquitectura_de_software

**ARQUITECTURA EN PIZARRA**

https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b

**ARQUITECTURA DE DESCOMPOSICIÓN MODULAR**

https://es.wikipedia.org/wiki/Arquitectura_de_software

## ***ROCÍO MÁRQUEZ MOLERO***

**ARQUITECTURA EN PIZARRA**

https://aprendearquitecturasoftware.wordpress.com/2018/10/05/patron-pizarra-blackboard-grupo-8/

**ARQUITECTURA DE CONTROL CENTRALIZADO**

https://www.hydraulicspneumatics.com/hp-en-espanol/article/21886630/4-razones-por-las-cuales-considerar-un-control-distribuido-vs-centralizado

**ARQUITECTURA DE CONTROL BASADA EN EVENTOS**

https://www.redhat.com/es/topics/integration/what-is-event-driven-architecture

## ***ÁLVERO CABA CARRILLO***

**ARQUITECTURA DE BUS DE EVENTOS**

https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b

**ARQUITECTURA INTERMEDIARIO**

https://blog.conasa.es/blog/patrones-comunes-de-arquitecturas-de-software

**ARQUITECTURA EN PIZARRA**

https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b

## ***MARTA LÓPEZ PÉREZ***

**ARQUITECTURA EN PIPELINE**

https://es.wikipedia.org/wiki/Arquitectura_en_pipeline_(inform%C3%A1tica)

**ARQUITECTURA DIRIGIDA POR EVENTOS**

https://apiumhub.com/es/tech-blog-barcelona/arquitectura-dirigida-por-eventos/

**ARQUITECTURA EN PIZARRA**

http://sistemascentradosendatos.blogspot.com/2013/10/arquitectura-de-pizarra.html

# ELECCIÓN GRUPAL

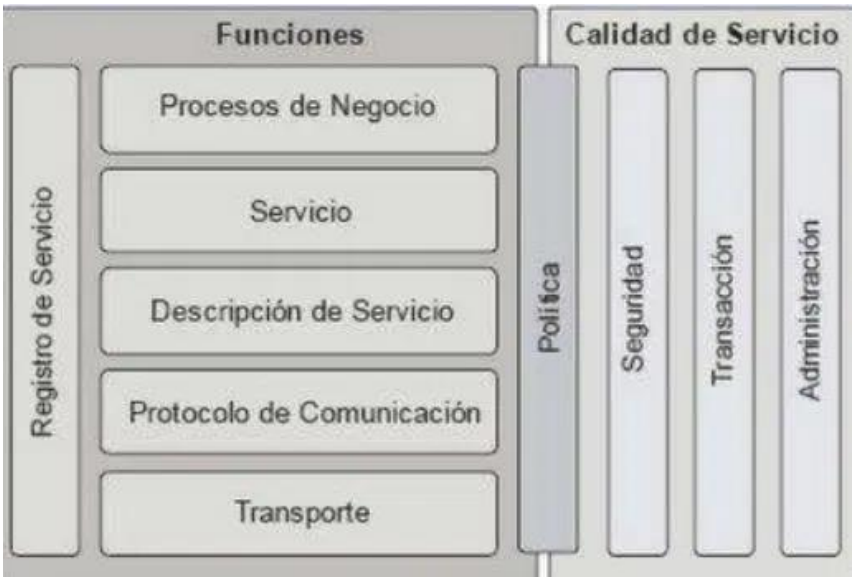
## ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

- ***Filosofía con grafico explicativo***

La Arquitectura SOA integra sistemas y aplicaciones heterogéneos en plataformas y protocolos de comunicación con metodologías bien definidas dentro de una serie de estándares empleados.

El propósito de SOA permite cambios posteriores en respuesta a los cambios en las necesidades de la empresa, aparte de crear una infraestructura integrada.

La activación y entrega de servicios parecen ser el núcleo de SOA, pero el valor real de esta arquitectura es la automatización de negocios. Por lo tanto, el enfoque en este tipo de arquitectura no debe enfocarse demasiado en los servicios, sino en los procesos y cómo mejorarlos.




- ***Descripción de sus componentes y sus conectores***

**COMPONENTES:**

**Servicio**: Función sin estado, autocontenida, que acepta una llamada y devuelve una respuesta mediante una interfaz bien definida.

**Cliente**: Consume el resultado del servicio provisto por un proveedor.

**CONECTORES:**

Antes, RPC, ahora Paso de mensajes.

- ***Ejemplos de aplicación***

1.- Cuando utilizamos nuestra tarjeta de débito para sacar dinero en el cajero automático.

2.- Cuando utilizamos nuestra tarjeta de crédito en algún establecimiento para adquirir bienes o servicios.

3.- Cuando utilizamos nuestro teléfono y el consumo de tiempo utilizado es facturado automáticamente en nuestro plan en tiempo real.

4.- Cuando pasamos por migración y se verifican nuestros datos contra los millones de registros de bases de datos de incidencia delictiva.

- ***Ventajas para nuestro proyecto***
  - Reduce el nivel de acoplamiento.
  - Clara definición de roles de desarrollo.
  - Definición de seguridad más clara.
  - Fácil testeo.
  - Mejora la mantención.
  - Favorece la reutilización.
  - Favorece el desarrollo en paralelo.
  - Permite fácil escalabilidad.
  - Permite un mapeo directo entre los procesos y los sistemas.
  - Permite un monitoreo preciso.
  - Permite la interoperabilidad.

- ***Otras consecuencias***

  Desventajas:
  - SOA depende de la implementación de estándares. Sin estándares, la comunicación entre aplicaciones requiere de mucho tiempo y código.
  - SOA no es para: aplicaciones con alto nivel de transferencia de datos, aplicaciones que no requieren de implementación del tipo request/response y para aplicaciones que tienen un corto periodo de vida.
  - Incrementalmente se hace difícil y costoso el ser capaz de cumplir con los protocolos y hablar con un servicio.
  - Implica conocer los procesos del negocio, clasificarlos, extraer las funciones que son comunes a ellos, estandarizarlas y formar con ellas capas de servicios que serán requeridas por cualquier proceso de negocio.
  - En la medida en que un servicio de negocio, vaya siendo incorporado en la definición de los procesos de negocio, dicho servicio aumentara su nivel de criticidad. Con lo cual cada que se requiera efectuar una actualización en dicho servicio, deberá evaluarse previamente el impacto y tener mucho cuidado con su implementación. Sin embargo, parte de la problemática anterior, puede ser solventada en virtud de un buen diseño del servicio.