

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy. X

# Los 10 patrones comunes de arquitectura de software

6 min read · Sep 7, 2018

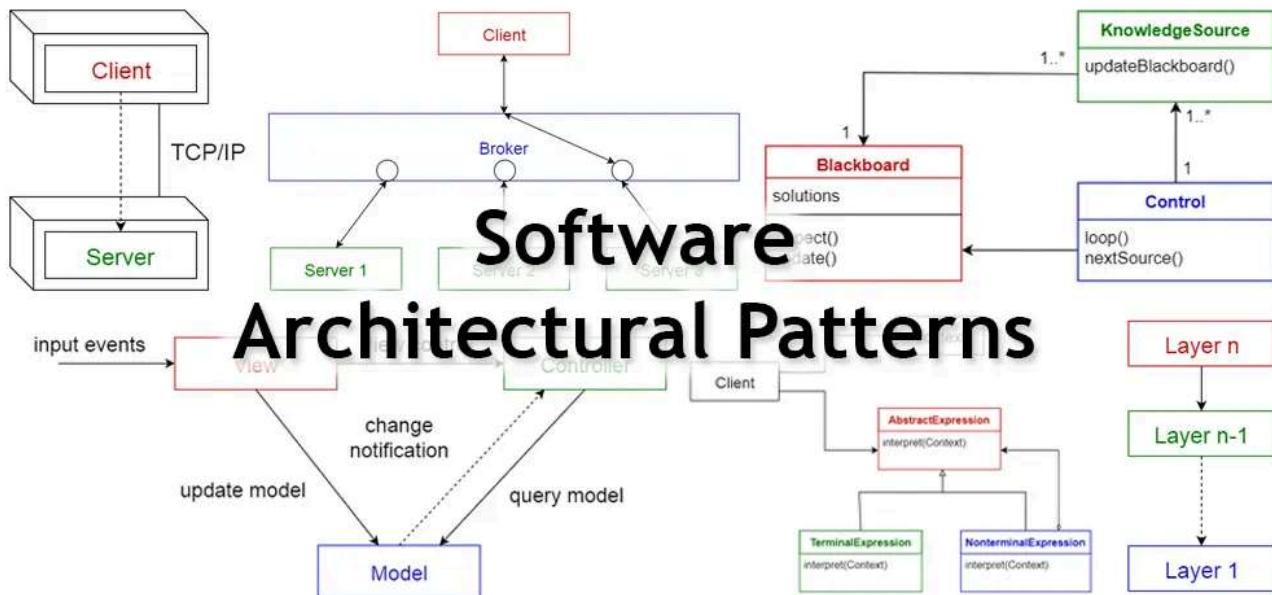


Wilber Ccori huaman

Follow

Share

¿Alguna vez se preguntó cómo se diseñan los grandes sistemas empresariales? Antes de que comience un importante desarrollo de software, debemos elegir una arquitectura adecuada que nos proporcione la funcionalidad deseada y los atributos de calidad. Por lo tanto, debemos entender diferentes arquitecturas, antes de aplicarlas a nuestro diseño.



## ¿Qué es un patrón arquitectónico?

De acuerdo con Wikipedia,

Un patrón arquitectónico es una solución general y reutilizable a un problema común en la arquitectura de software. To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

En este artículo, explicaré brevemente los siguientes 10 patrones arquitectónicos comunes con su uso, pros y contras.

1. Patrón de capas
2. Patrón cliente-servidor
3. Patrón maestro-esclavo
4. Patrón de filtro de tubería
5. Patrón de intermediario
6. Patrón de igual a igual
7. Patrón de bus de evento
8. Modelo-vista-controlador
9. Patrón de pizarra
10. Patrón de intérprete

## 1. Patrón de capas

Este patrón se puede utilizar para estructurar programas que se pueden descomponer en grupos de subtareas, cada una de las cuales se encuentra en un nivel particular de abstracción. Cada capa proporciona servicios a la siguiente capa superior.

[Open in app](#) ↗

[Sign up](#)

[Sign in](#)

# Medium



Search



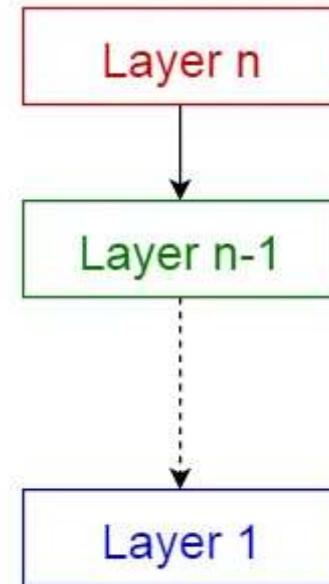
- **Capa de presentación** (también conocida como **capa UI**)
- **Capa de aplicación** (también conocida como **capa de servicio**)
- **Capa de lógica de negocios** (también conocida como **capa de dominio**)
- **Capa de acceso a datos** (también conocida como **capa de persistencia**)

## Uso

- Aplicaciones de escritorio generales

- Aplicaciones web

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



Patrón de capas

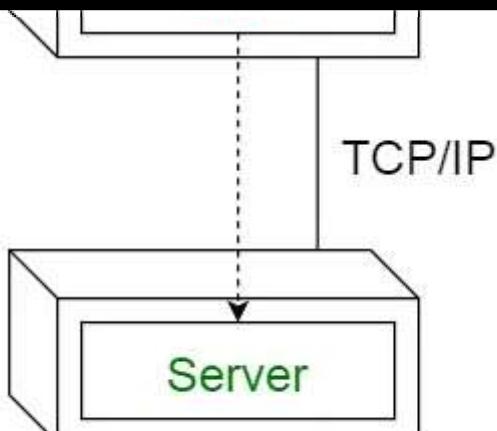
## 2. Patrón cliente-servidor

Este patrón consiste en dos partes; un **servidor** y múltiples **clientes**. El componente del servidor proporcionará servicios a múltiples componentes del cliente. Los clientes solicitan servicios del servidor y el servidor proporciona servicios relevantes a esos clientes. Además, el servidor sigue escuchando las solicitudes de los clientes.

### Uso

- Aplicaciones en línea como correo electrónico, uso compartido de documentos y banca.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



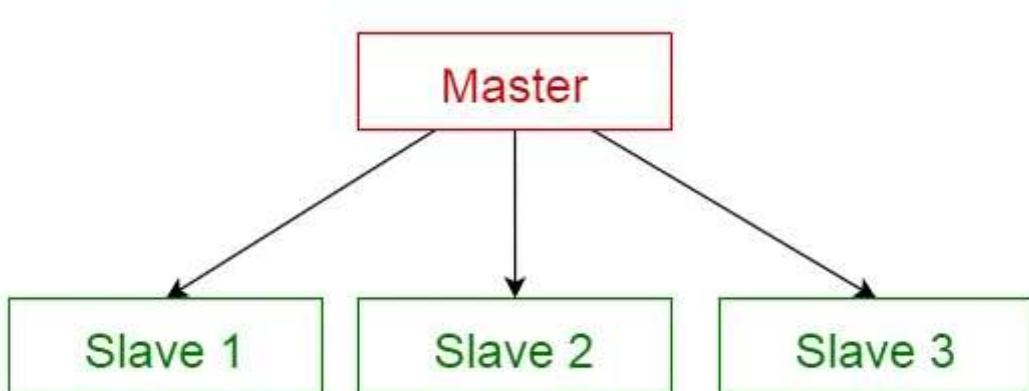
Patrón cliente-servidor

### 3. Patrón maestro-esclavo

Este patrón consiste en dos partes; **maestro** y **esclavos**. El componente maestro distribuye el trabajo entre componentes esclavos idénticos y calcula el resultado final de los resultados que devuelven los esclavos.

#### Uso

- En la replicación de la base de datos, la base de datos maestra se considera como la fuente autorizada y las bases de datos esclavas se sincronizan con ella.
- Periféricos conectados a un bus en un sistema informático (unidades maestra y esclava).



Patrón maestro-esclavo

### 4. Patrón de filtro de tubería

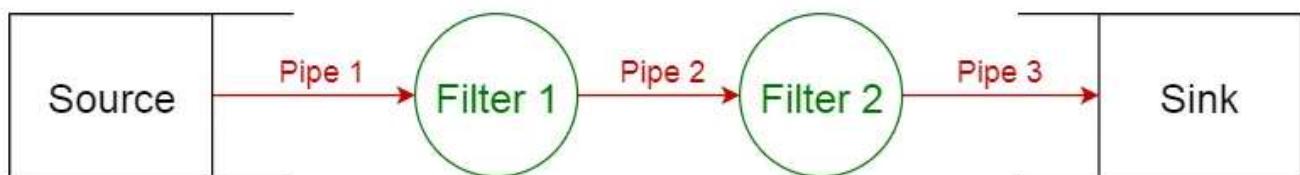
Este patrón se puede usar para estructurar sistemas que producen y procesan una secuencia de datos. Cada paso de procesamiento se incluye dentro de un

componente de filtro. Los datos que se procesarán se pasan a través de las tuberías

. Estas tuberías tienen fines de sincronización. To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

## Uso

- Compiladores Los filtros consecutivos realizan análisis léxico, análisis sintáctico y generación de código.
- Flujos de trabajo en bioinformática.



Patrón de filtro de tubería

## 5. Patrón del agente

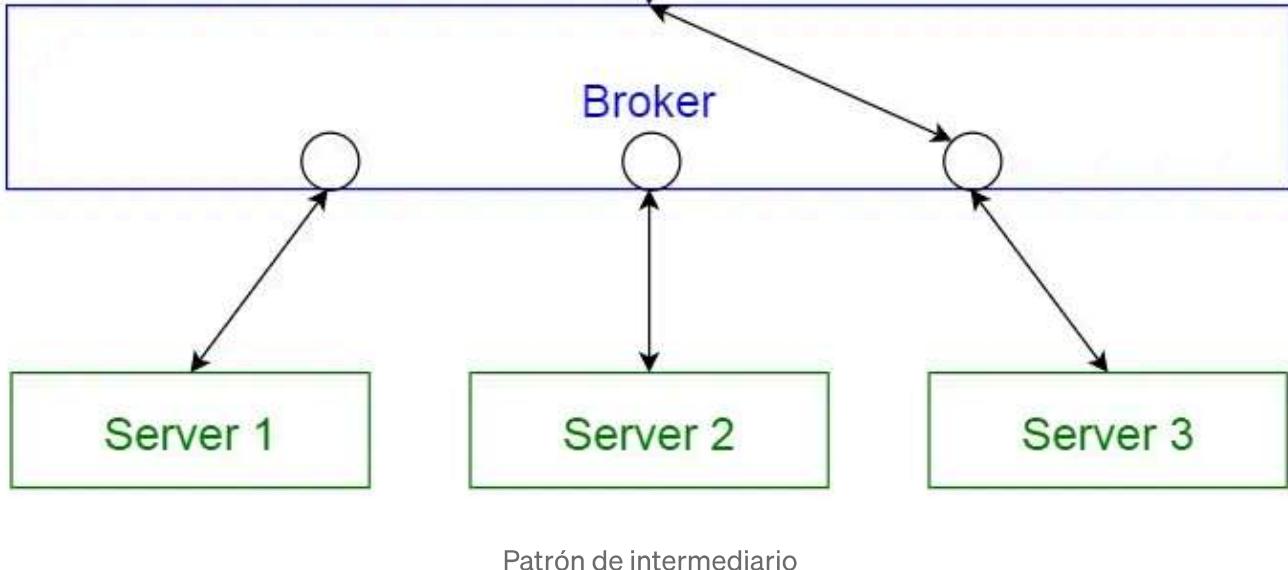
Este patrón se usa para estructurar sistemas distribuidos con componentes desacoplados. Estos componentes pueden interactuar entre sí mediante invocaciones de servicios remotos. Un componente de **intermediario** es responsable de la coordinación de la comunicación entre los componentes .

Los servidores publican sus capacidades (servicios y características) a un intermediario. Los clientes solicitan un servicio del intermediario y el intermediario redirecciona al cliente a un servicio adecuado desde su registro.

## Uso

- Software de Message Broker como [Apache ActiveMQ](#) , [Apache Kafka](#) , [RabbitMQ](#) y [JBoss Messaging](#) .

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



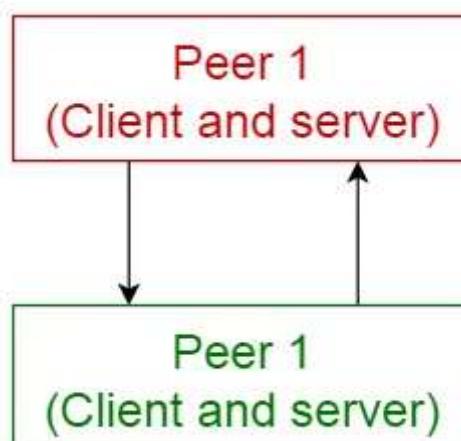
Patrón de intermediario

## 6. Patrón de igual a igual

En este patrón, los componentes individuales se conocen como **pares**. Los pares pueden funcionar tanto como un **cliente**, solicitando servicios de otros pares, y como un **servidor**, proporcionando servicios a otros pares. Un par puede actuar como un cliente o como un servidor o como ambos, y puede cambiar su rol dinámicamente con el tiempo.

### Uso

- Redes de intercambio de archivos como [Gnutella](#) y [G2](#) )
- Protocolos multimedia como [P2PTV](#) y [PDTP](#) .



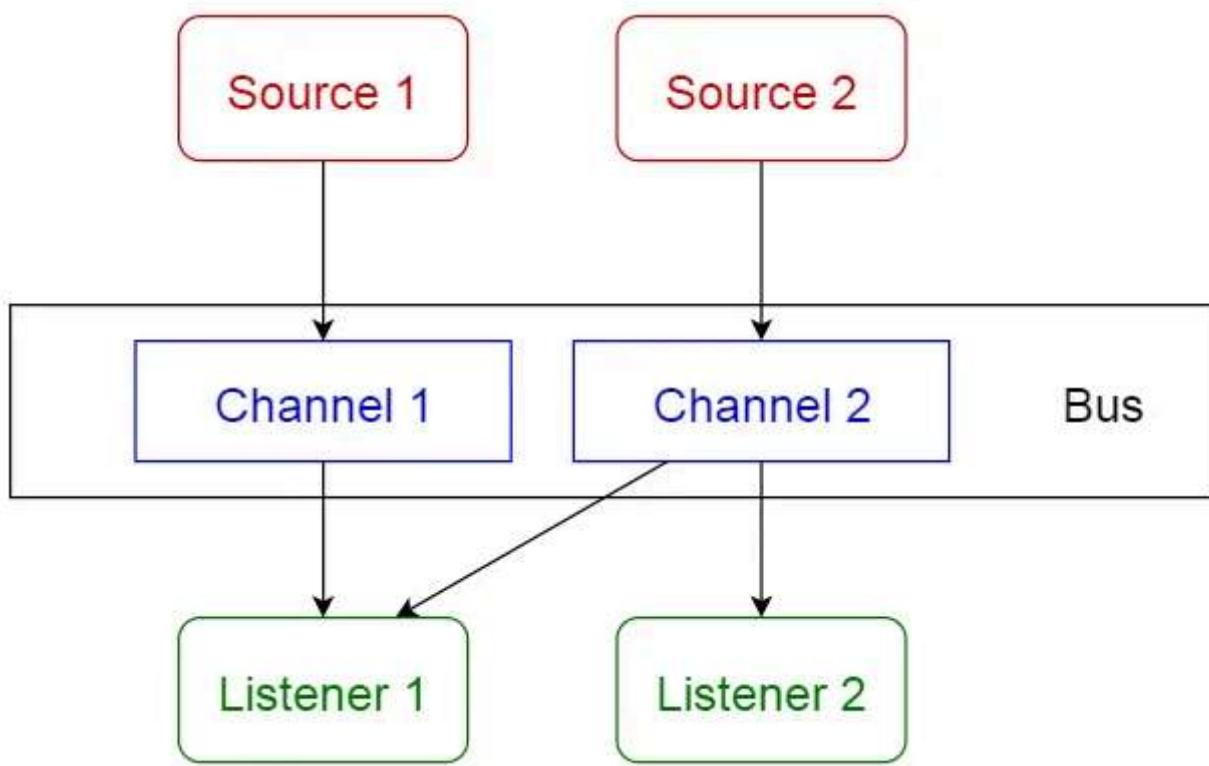
**Patrón de igual a igual****7. Patrón**

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Este patrón trata principalmente con eventos y tiene 4 componentes principales; fuente de evento , escucha de evento , canal y bus de evento . Las fuentes publican mensajes en canales particulares en un bus de eventos. Los oyentes se suscriben a canales particulares. Los oyentes son notificados de los mensajes que se publican en un canal al que se han suscrito anteriormente.

**Uso**

- Desarrollo de Android
- Servicios de notificación



Patrón de bus de evento

**8. Patrón de modelo-vista-controlador**

Este patrón, también conocido como patrón MVC, divide una aplicación interactiva en 3 partes, como

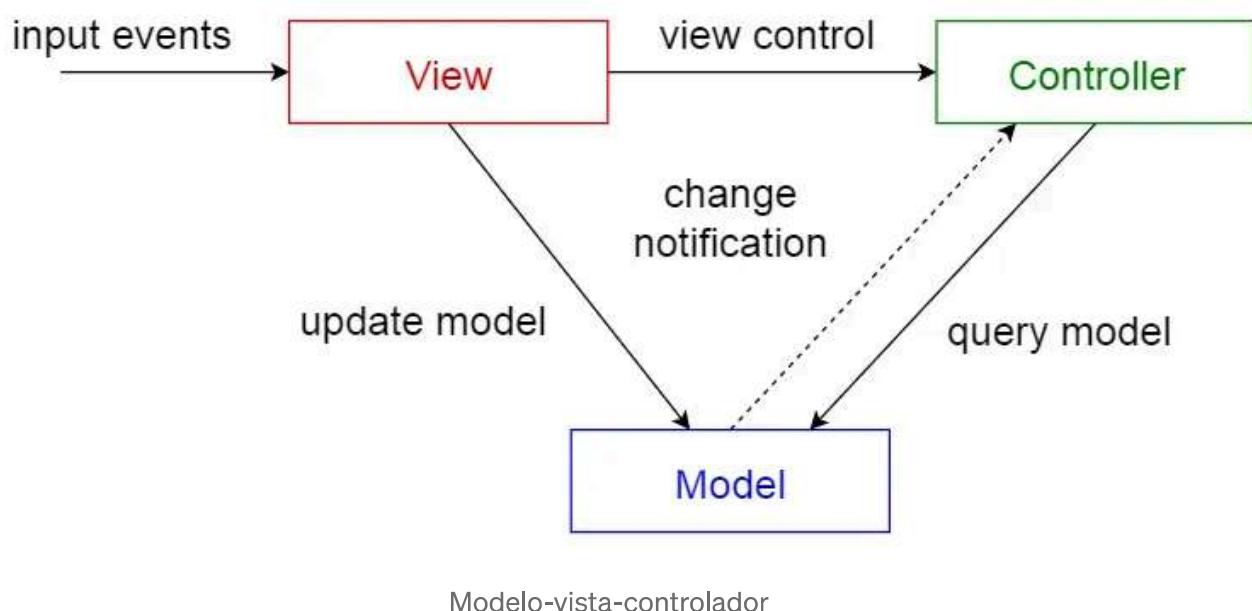
1. **modelo** — contiene la funcionalidad y los datos básicos
2. **vista** : muestra la información al usuario (se puede definir más de una vista)
3. **controlador** : maneja la entrada del usuario

Esto se hace para separar las representaciones internas de información de las formas en que se presentan a los usuarios.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

## Uso

- Arquitectura para aplicaciones World Wide Web en los principales lenguajes de programación.
- Marcos web como [Django](#) y [Rails](#).



## 9. Patrón de pizarra

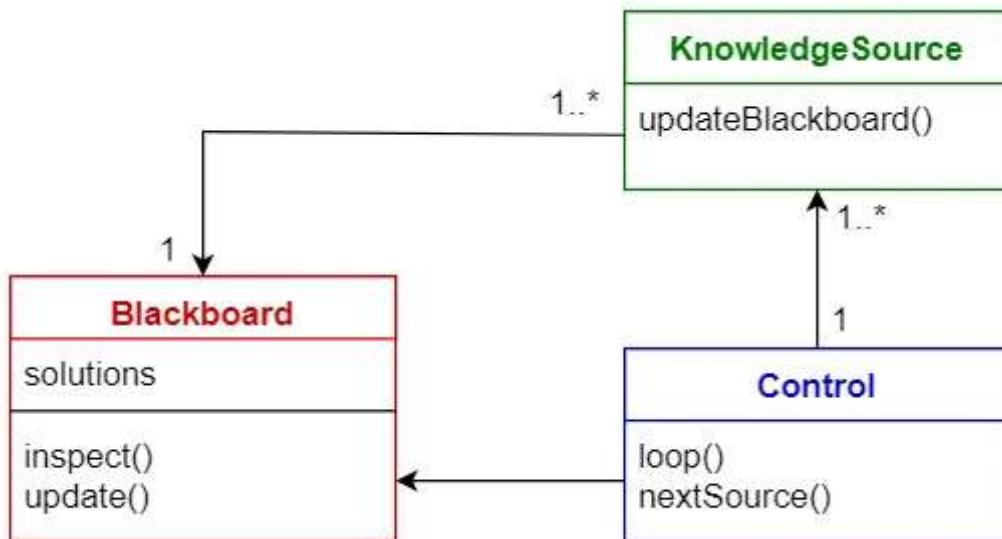
Este patrón es útil para problemas para los que no se conocen estrategias de solución deterministas. El patrón de pizarra consta de 3 componentes principales.

- **pizarra**: una memoria global estructurada que contiene objetos del espacio de solución
- **fuente de conocimiento**: módulos especializados con su propia representación
- **componente de control**: selecciona, configura y ejecuta módulos.

Todos los componentes tienen acceso a la pizarra. Los componentes pueden producir nuevos objetos de datos que se agregan a la pizarra. Los componentes buscan tipos particulares de datos en la pizarra, y pueden encontrarlos por coincidencia de patrones con la fuente de conocimiento existente.

## Uso

- Reconocimiento de voz
- Identificación de la estructura proteica
- Sonar señala la interpretación.



Patrón de pizarra

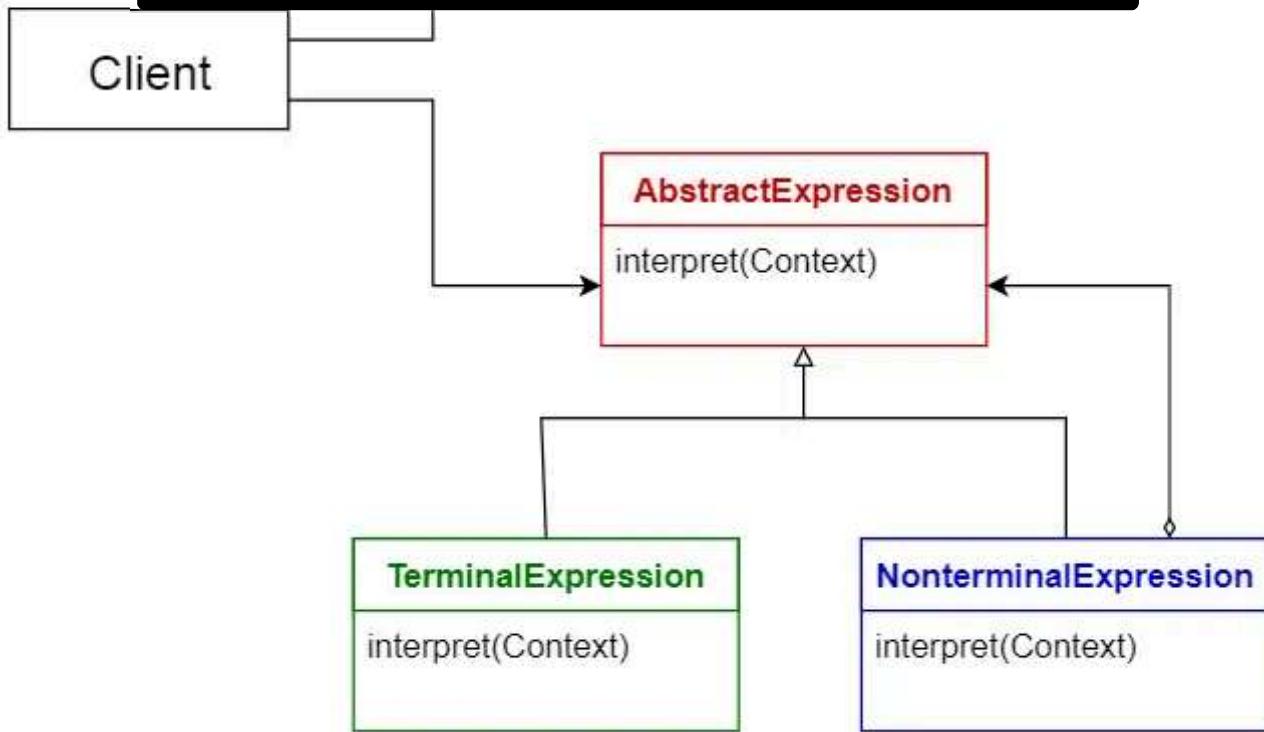
## 10. Patrón de intérprete

Este patrón se usa para diseñar un componente que interpreta programas escritos en un lenguaje dedicado. Especifica principalmente cómo evaluar las líneas de programas, conocidas como oraciones o expresiones escritas en un idioma particular. La idea básica es tener una clase para cada símbolo del idioma.

### Uso

- Lenguajes de consulta de base de datos como SQL.
- Idiomas utilizados para describir los protocolos de comunicación.

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



Patrón de intérprete

## Comparación de patrones arquitectónicos

La tabla que figura a continuación resume los pros y los contras de cada patrón arquitectónico.

Name	Advantages	Disadvantages
Layered	A lower layer can be used by different higher layers. Layers make standardization easier as we can clearly define levels. Changes can be made within the layer without affecting other layers.	Not universally applicable. Certain layers may have to be skipped in certain situations.
Client-server	Good to model a set of services where clients can request them.	Requests are typically handled in separate threads on the server. Inter-process communication causes overhead as different clients have different representations.
Master-slave	Accuracy - The execution of a service is delegated to different slaves, with different implementations.	The slaves are isolated: there is no shared state. The latency in the master-slave communication can be an issue, for instance in real-time systems. This pattern can only be applied to a problem that can be decomposed.
Pipe-filter	Exhibits concurrent processing. When input and output consist of streams, and filters start computing when they receive data. Easy to add filters. The system can be extended easily. Filters are reusable. Can build different pipelines by recombining a given set of filters	Efficiency is limited by the slowest filter process. Data-transformation overhead when moving from one filter to another.
Broker	Allows dynamic change, addition, deletion and relocation of objects, and it makes distribution transparent to the developer.	Requires standardization of service descriptions.
Peer-to-peer	Supports decentralized computing. Highly robust in the failure of any given node. Highly scalable in terms of resources and computing power.	There is no guarantee about quality of service, as nodes cooperate voluntarily. Security is difficult to be guaranteed. Performance depends on the number of nodes.
Event-bus	New publishers, subscribers and connections can be added easily. Effective for highly distributed applications.	Scalability may be a problem, as all messages travel through the same event bus
Model-view-controller	Makes it easy to have multiple views of the same model, which can be connected and disconnected at run-time.	Increases complexity. May lead to many unnecessary updates for user actions.
Blackboard	Easy to add new applications. Extending the structure of the data space is easy.	Modifying the structure of the data space is hard, as all applications are affected. May need synchronization and access control.
Interpreter	Highly dynamic behavior is possible. Good for end user programmability. Enhances flexibility, because replacing an interpreted program is easy.	Because an interpreted language is generally slower than a compiled one, performance may be an issue.

## Comparación de patrones arquitectónicos

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

Espero que te gusten mis  
pensamientos 😊

Gracias por leer. 😊

¡Aclamaciones! 😊

- [Arquitectura de software](#)
- [Programación](#)

Me gusta lo que lees? Dale una ronda de aplausos.

Desde un aplauso rápido hasta una ovación de pie, aplauda para mostrar cuánto disfrutaste de esta historia.

Para conocer mas tecnologias pueden seguirme en mis redes sociales:

Facebook: <https://web.facebook.com/wilbeDEV/>

facebook: <https://web.facebook.com/TechManiako/>

Grupo\_facebook: <https://web.facebook.com/groups/166091130742535/>

YouTube: <https://www.youtube.com/c/WILBERCCORIDEVELOPER>

Blogs: <https://www.techmaniako.com/ED/>

GitHub: <https://github.com/wilberCcoriHuman>



Follow

**Written by Wilber Ccori huaman**

173 followers • 3 following

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



## Responses



Write a response

What are your thoughts?



Betania Beltrán

Mar 21, 2019

...

ESTE ARTÍCULO ES UN PLAGIO, YA HABÍA SIDO PUBLICADO 1 AÑO ANTES POR OTRA PERSONA Y SE PUEDE VER EN EL SIGUIENTE LINK — — <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>



13

[Reply](#)



María del Pilar Rius Hernández

Jan 19, 2020

...

Muy buen trabajo Wilber, muchas gracias por compartir!! Saludos



1

[Reply](#)



Nicolás Rubén Caballero Ortega

Jan 18, 2022

...

Considera enlazar el artículo del cual te basaste. Ese tiene la referencia al documento original, con los textos que explican cada uno de los patrones.



[Reply](#)

## More from Wilber Ccori huaman



Wilber Ccori huaman

## ¿Qué es un autómata?

Un autómata es un modelo matemático para una máquina de estado finito, en el que dada una entrada de símbolos, “salta” mediante una serie...

Aug 28, 2018

17



# SQL



Wilber Ccori huaman

## SQL BÁSICO: CONCEPTOS BÁSICOS

SQL (Structured Query Language) o en español Lenguaje Estructurado de Consulta es el lenguaje utilizado para definir, controlar y acceder a...

Aug 28, 2018

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



Wilber Ccori huaman

## 10 Técnicas de estimación de software

¿Te gustaría formarte en técnicas para estimar exactitud los proyectos de software, cumplir tus metas de tiempo, costo y satisfacer las...

Sep 7, 2018

3



Wilber Ccori huaman

## ¿Qué es RPC (llamada a procedimiento remoto)?

Llamada a p  
una técnica

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.

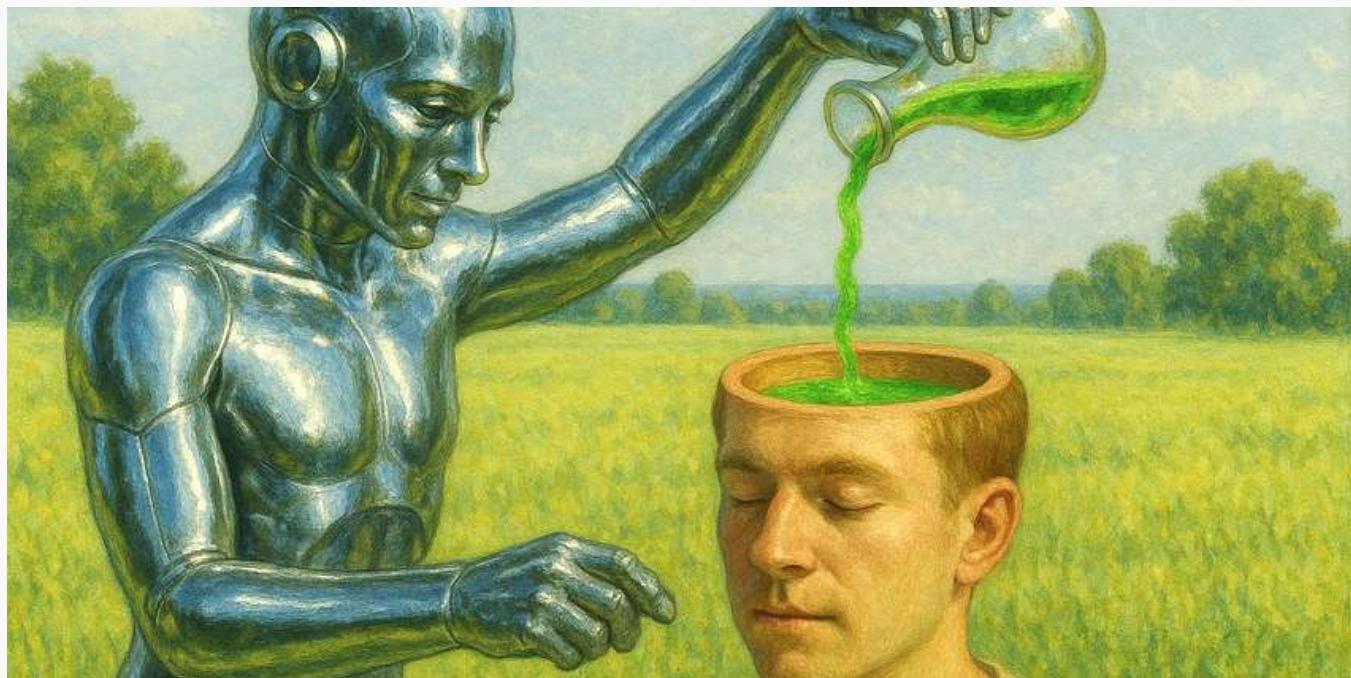
Aug 28, 2018

31



See all from Wilber Ccori huaman

## Recommended from Medium



Jordan Gibbs

### ChatGPT Is Poisoning Your Brain...

Here's How to Stop It Before It's Too Late.

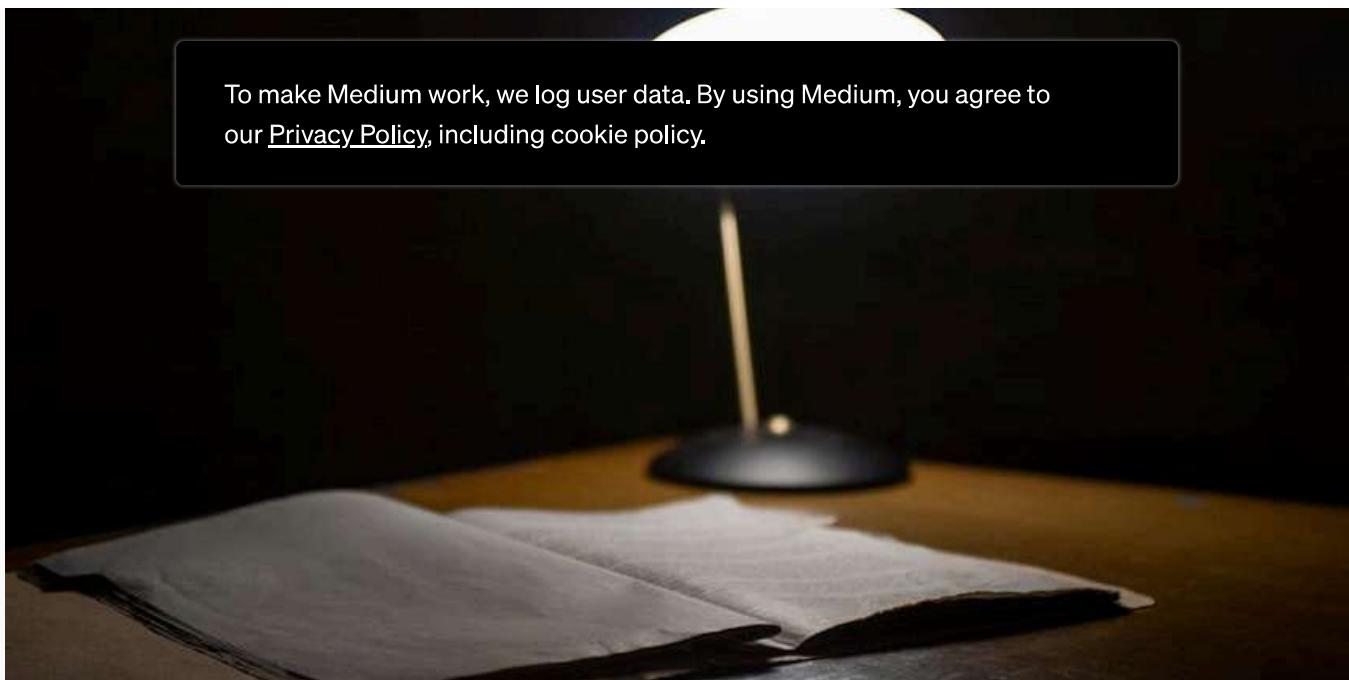
♦ Apr 30

19.5K

898



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



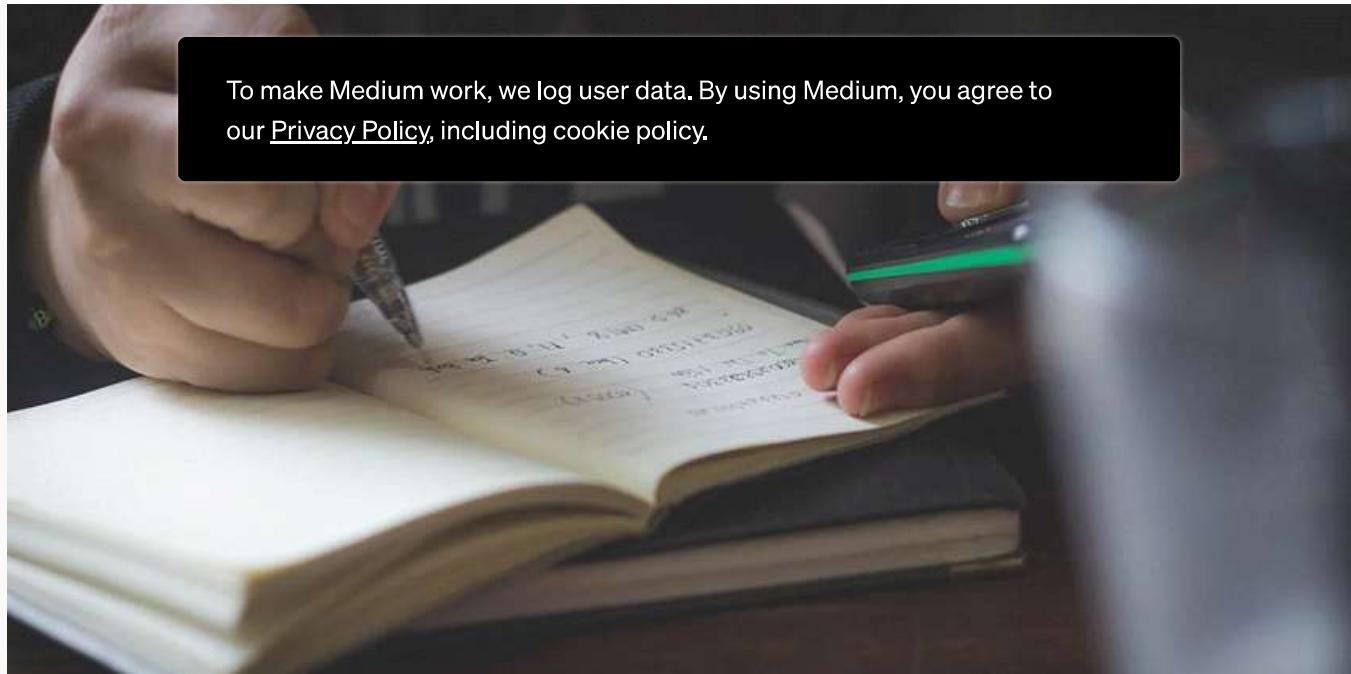
LONG In Long. Sweet. Valuable. by Ossai Chinedum

## I'll Instantly Know You Used Chat Gpt If I See This

Trust me you're not as slick as you think

★ May 16 ⌘ 10.3K 🗣 568





In Write A Catalyst by Adarsh Gupta

## How I Study Consistently While Working a 9–5 Full-Time Job

No, I don't wake up at 5 AM. And yes, I have a life.

4 Apr 21 6.2K 265



In Psychology of Workplaces by George J. Ziegas

## Resumes Are Dying—Here's What's Replacing Them

How modern hiring is leaving resumes behind

4 Jun 9 13K 370



To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



In ThinkDraft by Singh Bhai

## How to Read Someone's Personality in 10 Seconds (Backed by Psychology)

The Subtle Signs That Reveal Who Someone Really Is.

Jan 27 18K 487



```

lib/main.dart
57 _MyHomePageState extends State<MyHomePage> {
58   int _counter = 0;
59
60   void _incrementCounter() {
61     setState(() {
62       _counter++;
63     });
64   }
65
66   @override
67   Widget build(BuildContext context) {
68     return Scaffold(
69       appBar: AppBar(
70         title: Text('Flutter Demo Home Page'),
71       ),
72       body: Center(
73         child: Column(
74           mainAxisAlignment: MainAxisAlignment.center,
75           children: <Widget>{
76             Text(
77               'You have pushed the button this many times:',
78             ),
79             Text(
80               '$_counter',
81               style: Theme.of(context).textTheme.headlineMedium,
82             ),
83           },
84         ),
85       ),
86       floatingActionButton: FloatingActionButton(
87         onPressed: _incrementCounter,
88         tooltip: 'Increment',
89         child: const Icon(Icons.add),
90       ),
91     );
92   }
93 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2025-03-02T14:37:39Z [Android] running flutter task "assembleDebug"...
2025-03-02T14:37:39Z [Android] Built build/app/outputs/flutter-apk/app-debug.apk
2025-03-02T14:37:39Z [Android] Lost connection to device.
2025-03-02T14:37:43Z [Android] Syncing files to device sdk gphone64 x86 64...
2025-03-02T14:37:43Z [Android] <IO> Preview running
2025-03-02T14:37:43Z [Android] I/Choreographer( 7793): Skipped 124 frames! The application may be doing too much work on its main thread.
2025-03-02T14:37:43Z [Android] I/Gralloc4( 7793): mapper 4.x is not supported
2025-03-02T14:37:43Z [Android] W/OpenGLRenderer( 7793): Failed to initialize 101010-2 format, error = EGL\_SUCCESS
2025-03-02T14:37:44Z [Android] I/Choreographer( 7793): Skipped 50 frames! The application may be doing too much work on its main thread.
2025-03-02T14:37:48Z [Android] D/ProfileInstaller( 7793): Installing profile for com.example.flutter

> OUTLINE > TIMELINE

In Coding Beauty by Tari Ibaba

## This new IDE from Google is an absolute game changer

This new IDE from Google is seriously revolutionary.

 Mar 11

To make Medium work, we log user data. By using Medium, you agree to our [Privacy Policy](#), including cookie policy.



See more recommendations