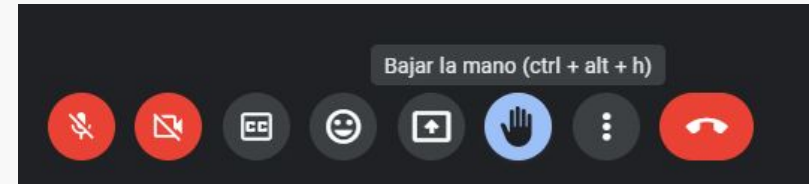
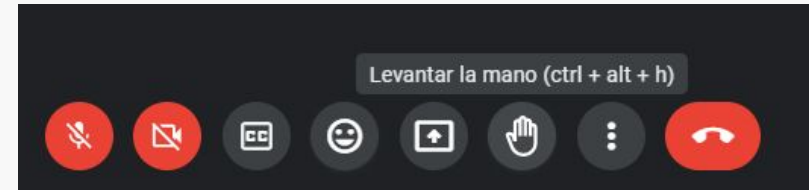
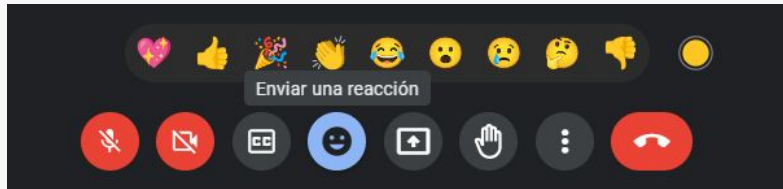


**El workshop será grabado y transmitido por Twitch.**

**Además, luego será subido a Youtube para que lo puedan volver a ver (junto con la presentación)**

## Algunas funcionalidades de Meet:



| WORKSHOP

# Entendiendo las bases de la arquitectura de software

2023 | by Webstarted



ACADEMY



# Speaker: Leonardo Guerberg

Soy Staff Engineer en **zerf**

---

**Vivo en** Capital Federal, Buenos Aires

---

**Soy Ingeniero en Sistemas de Información**

Me recibí en Agosto de 2023 de la Universidad  
Tecnológica Nacional.

**Entonces hoy vamos a  
discutir arquitecturas y  
conceptos súper complejos?**



# Contenido

**1**

Qué implica pensar en arquitectura y por qué es necesario

**2**

Qué es la arquitectura de software

**3**

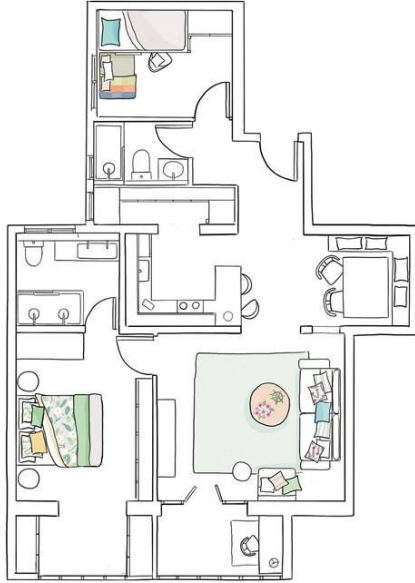
El rol del arquitecto

**5**

Principios SOLID

**4**

Relación diseño y arquitectura de software



# QUE IMPLICA HACER ARQUITECTURA

**Pensemos en construir una casa**

Antes de empezar a **construir** las habitaciones, **pintar** las paredes o **instalar** las ventanas, primero necesitas un plan.





# DISEÑO INICIAL Y ESTRUCTURA DE UNA APLICACIÓN

Funciona como un mapa o guía constante para los desarrolladores a la hora de construir software **eficiente** y **escalable**.

Existen varios aspectos que hay que tener en cuenta a la hora de pensar en **arquitectura**, repasemos algunos manteniendo la analogía de la casa...

# ESCALABILIDAD

## ➔ Nuevas funcionalidades

Poder agregar habitaciones nuevas sin romper la casa.

## ➔ Uso de recursos

La cantidad de ladrillos y cemento es limitada.

## ➔ Organización de tareas

Múltiples obreros deben poder trabajar al mismo tiempo sin afectar al otro.

## COMPONENTES Y CONEXIONES

➔ **Protocolos de comunicación**  
Pasillos de la casa.

➔ **Definición de responsabilidades**  
Cada habitación tiene su objetivo.

➔ **Acoplamiento y dependencias**  
Pensar si hay habitaciones que dependen de otras.

# MANTENIBILIDAD



## Fix de errores

Poder reparar una habitación sin romper otra.

# SEGURIDAD



## Proteger los recursos

Vos podrías hacer una casa sin llave en la puerta, hasta que alguien se de cuenta...

# RESILIENCIA



## Recupero ante fallas

Todos los sistemas fallan, pero deben ser capaz de recuperarse al volver a un estado consistente y funcional.



La arquitectura de software es la **planificación** y el **diseño estructural** de un programa, asegurando que esté bien **organizado, escalable, fácil de mantener, eficiente y seguro.**

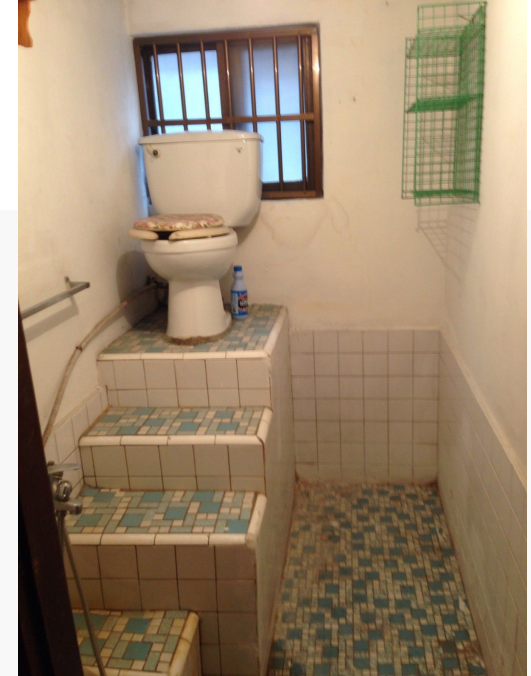
---

**El esfuerzo que lleva mantener y construir un sistema es inversamente proporcional a la calidad de su arquitectura.**



# ¿QUÉ ES LA ARQUITECTURA DE SOFTWARE?

Vamos a comparar estas imágenes...



# CASOS DE USO

Si es un sistema estilo marketplace, la arquitectura debe **soportar y reflejar** el mismo (usuarios, carritos de compra, búsqueda de productos, etc...)



## SCREAMING ARCHITECTURE

# OPERATIVA



Si mi sistema está pensado para soportar **miles de peticiones**, la arquitectura debe soportarlo.

# DESPLIEGUE

Una buena arquitectura permite un sistema ser desplegado a un entorno productivo de manera **fácil y sencilla y sin requerir exceso de esfuerzo manual.**





# EL ROL DE ARQUITECTO

1. **Programador**, y de los mejores.
2. Por más que **no programe tanto como uno**, nunca puede dejar de programar.
3. Busca **entender problemas** para dar soluciones que se abstraiga de los detalles de la implementación.
4. Posee suficiente **criterio** para estar constantemente tomando los mejores **trade-offs**.
5. Es parte del **camino de crecimiento** enfocado en lo técnico dentro del desarrollo.



# ARQUITECTURA Y DISEÑO

Son **prácticamente lo mismo** si nos basamos en el día a día de un arquitecto de software.

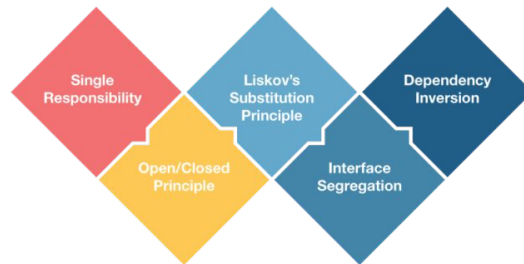
Arquitectura implica decisiones más de **alto nivel** (como fueran los planos de la casa a nivel distribución de la propiedad) mientras que diseño son decisiones **mas chicas pero que impactan también** en todos o alguno de los puntos nombrados anteriormente (como la conexión de luz de un dormitorio).



# PRINCIPIOS SOLID

Son **reglas** o **guías** de diseño de software que buscan crear código más comprensible, flexible y mantenible.

## S.O.L.I.D.



# SINGLE RESPONSIBILITY

Cada módulo, clase o función debería tener una **única razón** para ser modificada.





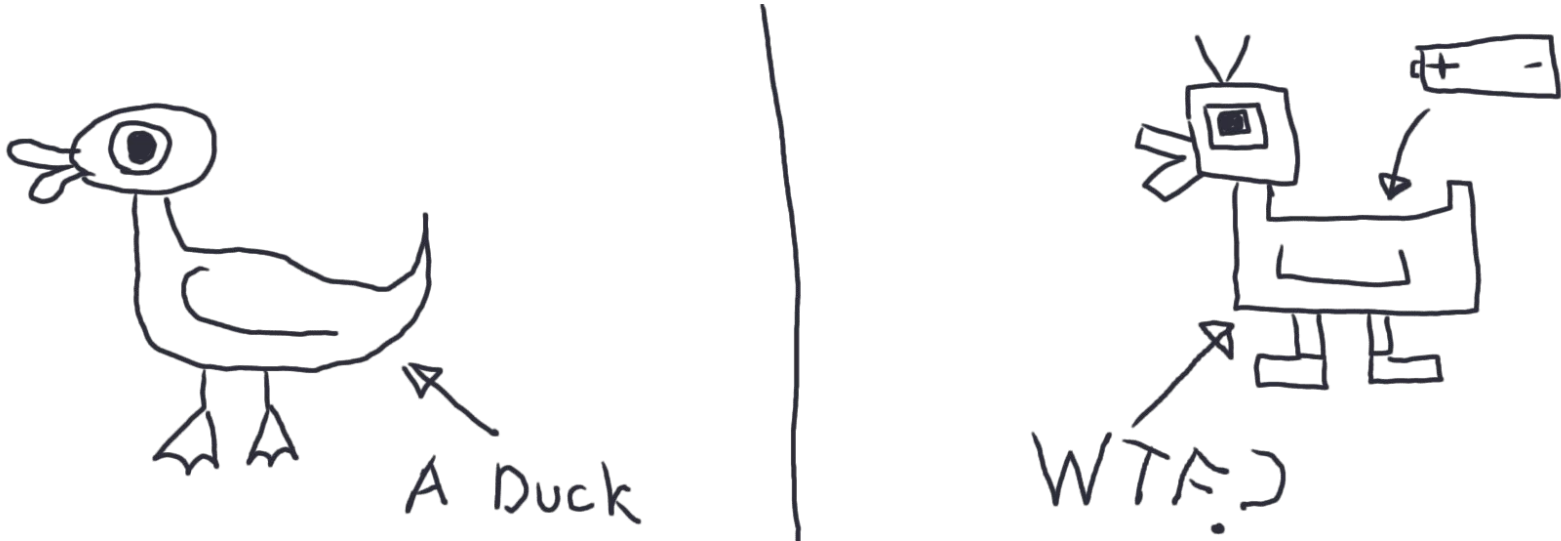
# OPEN/CLOSED

Tu código debería permitir la **adición** de nuevas funcionalidades sin **cambiar** el código existente.



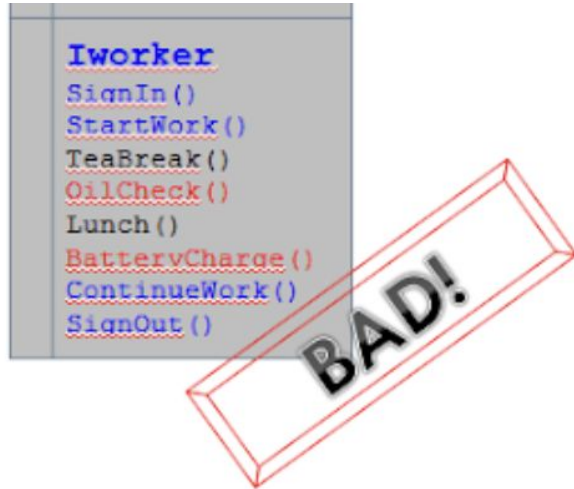
# LISKOV SUBSTITUTION

Los componentes bases de mi sistema deben ser **intercambiables** sin afectar la funcionalidad del programa. Se enfoca en crear abstracciones para estos componentes.



# INTERFACE SEGREGATION

No debes forzar a las clases a implementar interfaces que no necesitan.



Iworker has methods  
that are different for  
different workers and  
violates ISP

Human

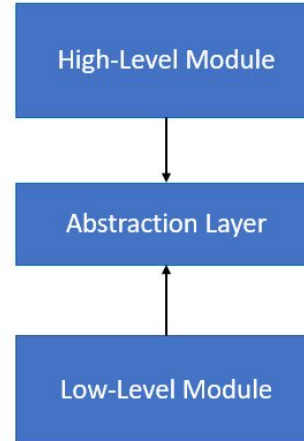
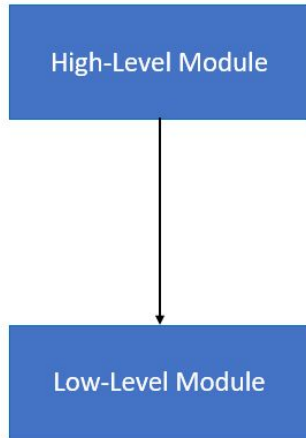


Robot



# DEPENDENCY INVERSION

Los módulos de alto nivel no deben depender de módulos de bajo nivel.





**FIGHT FOR THE ARCHITECTURE**



**zerf**

**¡GRACIAS!**



ACADEMY



**¡Seguinos en nuestras redes y activa las notificaciones para recibir novedades!**



CONOCÉ NUESTRAS PROPUESTAS EN  
[www.webstartedacademy.com](http://www.webstartedacademy.com)



Esta iniciativa es patrocinada por  webstarted

## We build distributed IT teams

Si querés armar tu propio equipo IT o conseguir un trabajo remoto  
ingresá a [www.webstarted.com](https://www.webstarted.com)



**Todas nuestras iniciativas,  
son 100% gratuita  
para todos los participantes.**

SI DESEAS CONTRIBUIR Y AYUDARNOS A SEGUIR OFRECIENDO ESTAS OPORTUNIDADES, TE PRESENTAMOS VARIAS OPCIONES PARA COLABORAR.

→ **DONACIÓN DE UNA SOLA VEZ (ARS)**  
<https://buy.stripe.com/bLYbL32Mm26lgSs5kk>

→ **SUSCRIPCIÓN MENSUAL DE 1 USD**  
<https://buy.stripe.com/cN23exaeObHi6dO147>

→ **DONACIÓN DE UNA SOLA VEZ (USD)**  
<https://buy.stripe.com/9AQ5mF72CaDeeKk3ce>

→ **SUSCRIPCIÓN MENSUAL DE 5 USD**  
<https://buy.stripe.com/bLYeXffz8cLm7hS8wA>

→ **DIFUNDIENDO LA INICIATIVA:**

**¡Gratis!**