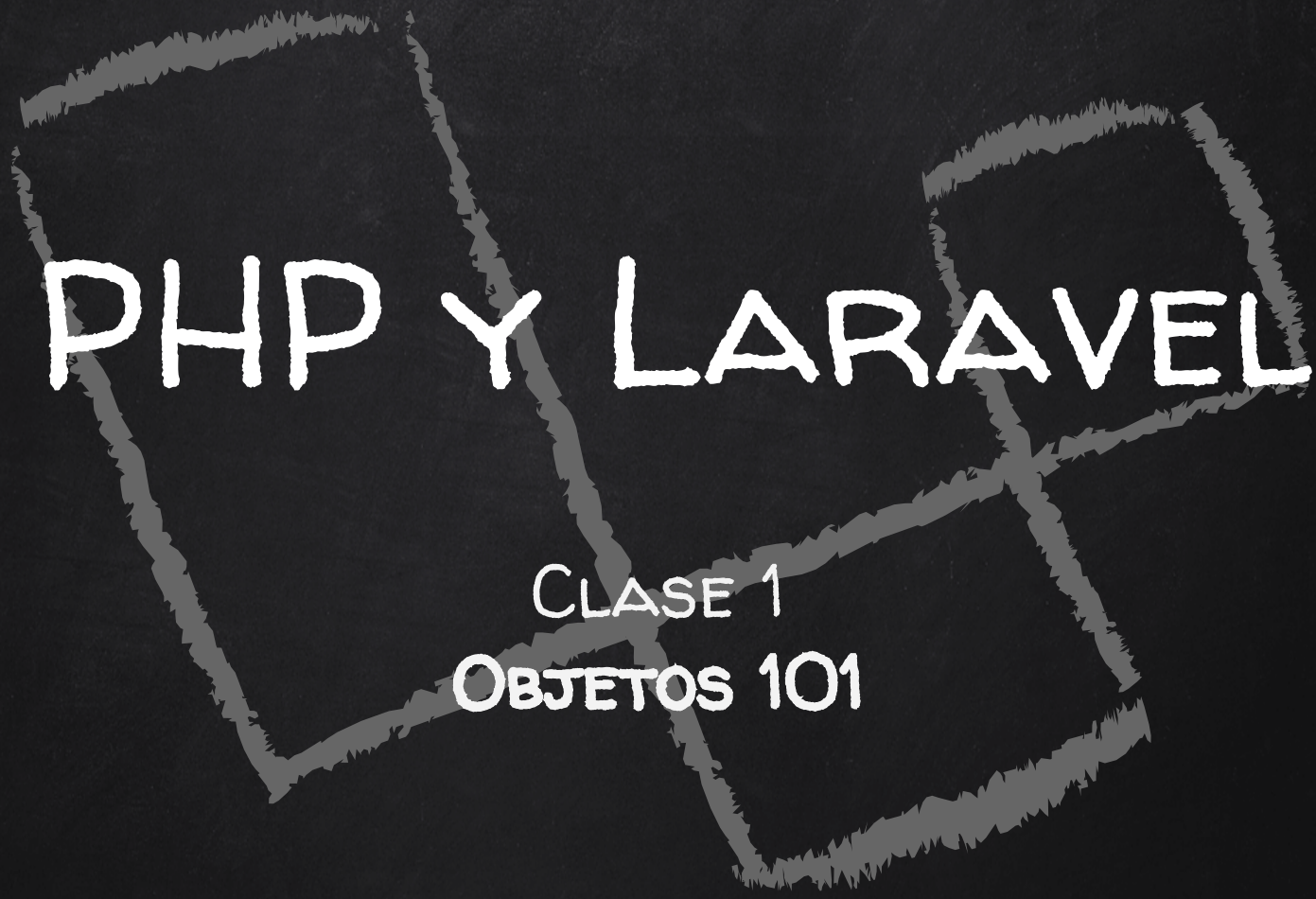
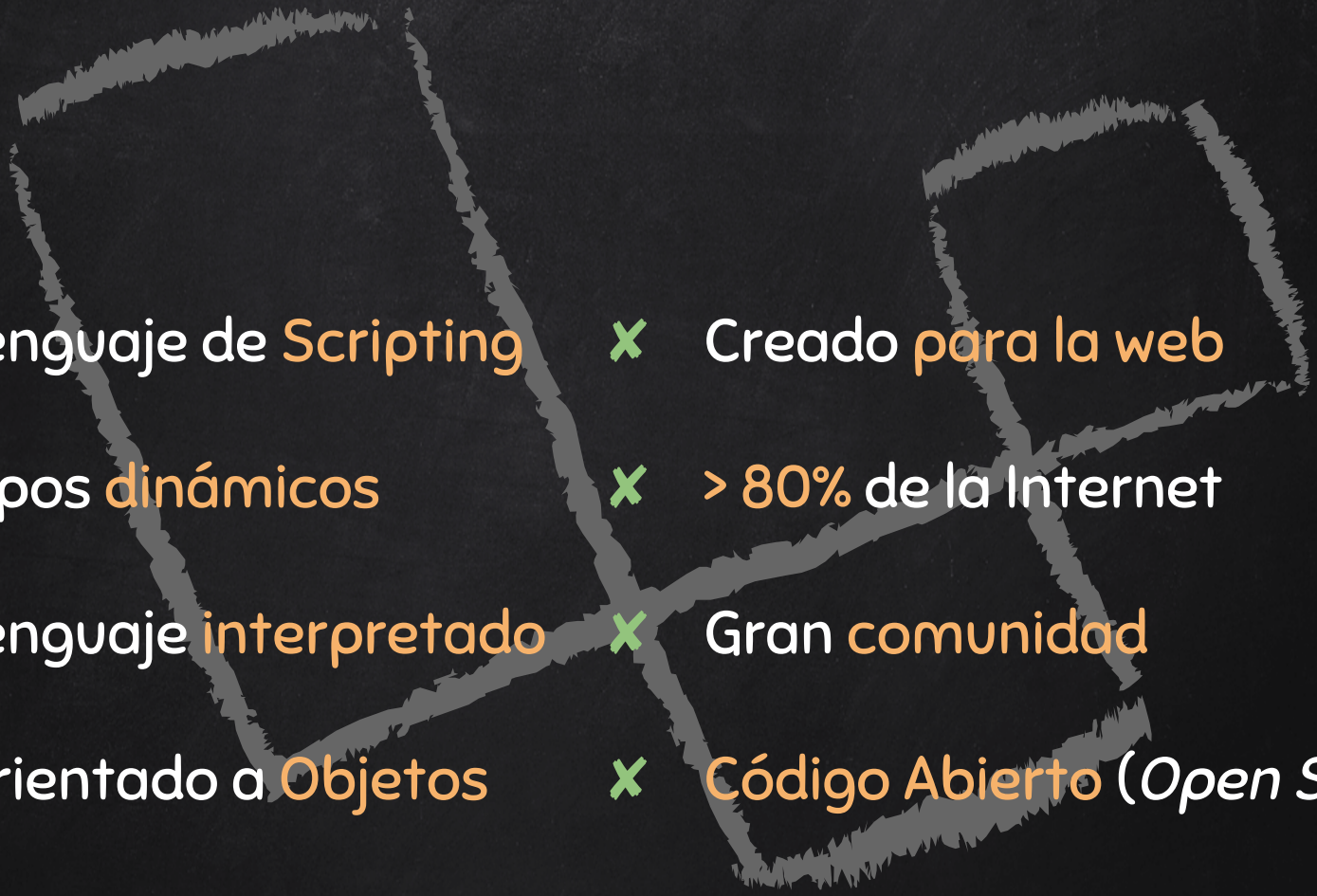


PHP y Laravel

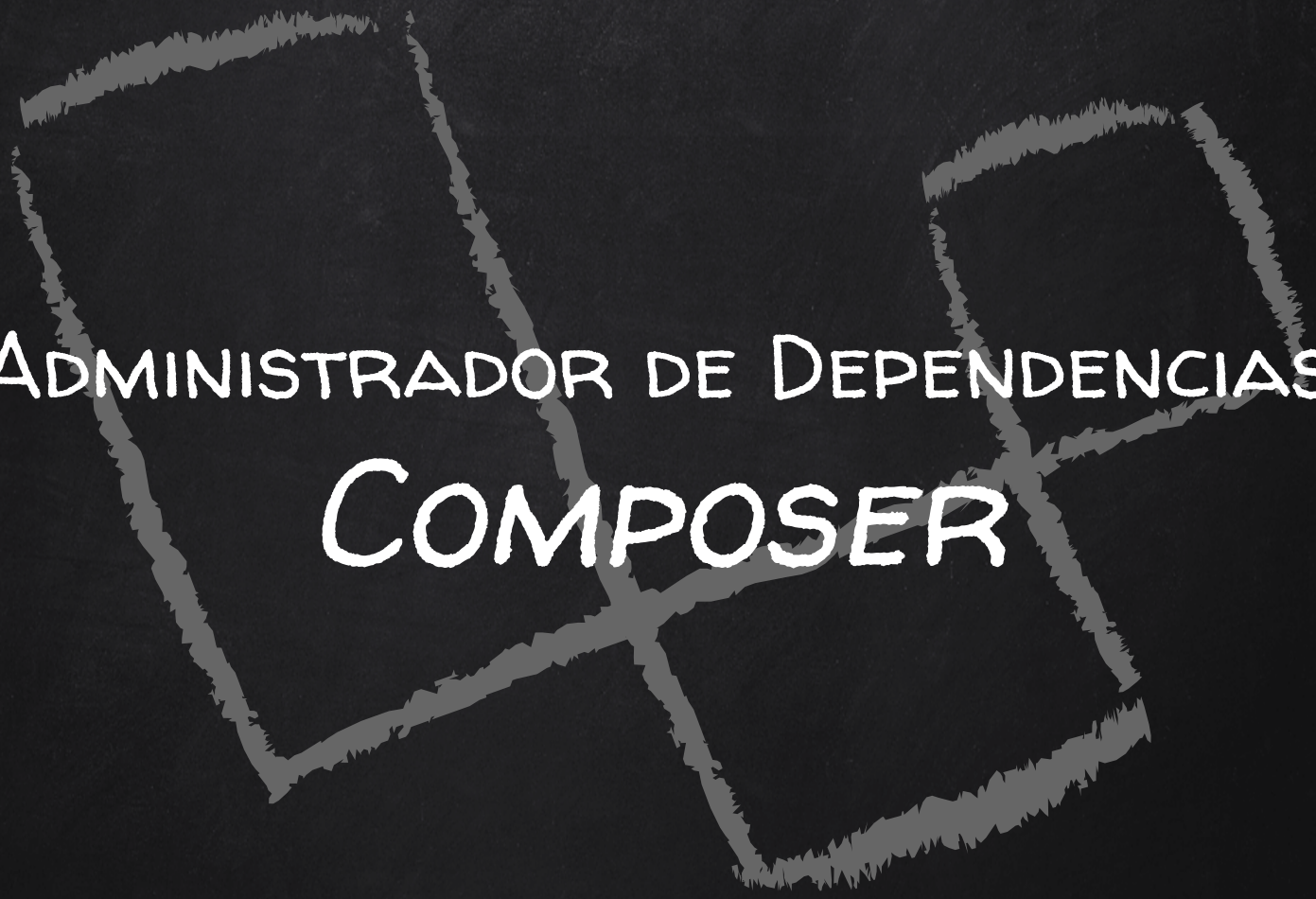
Clase 1
Objetos 101

Guido Contreras Woda – @guiwoda

¿Qué es PHP?



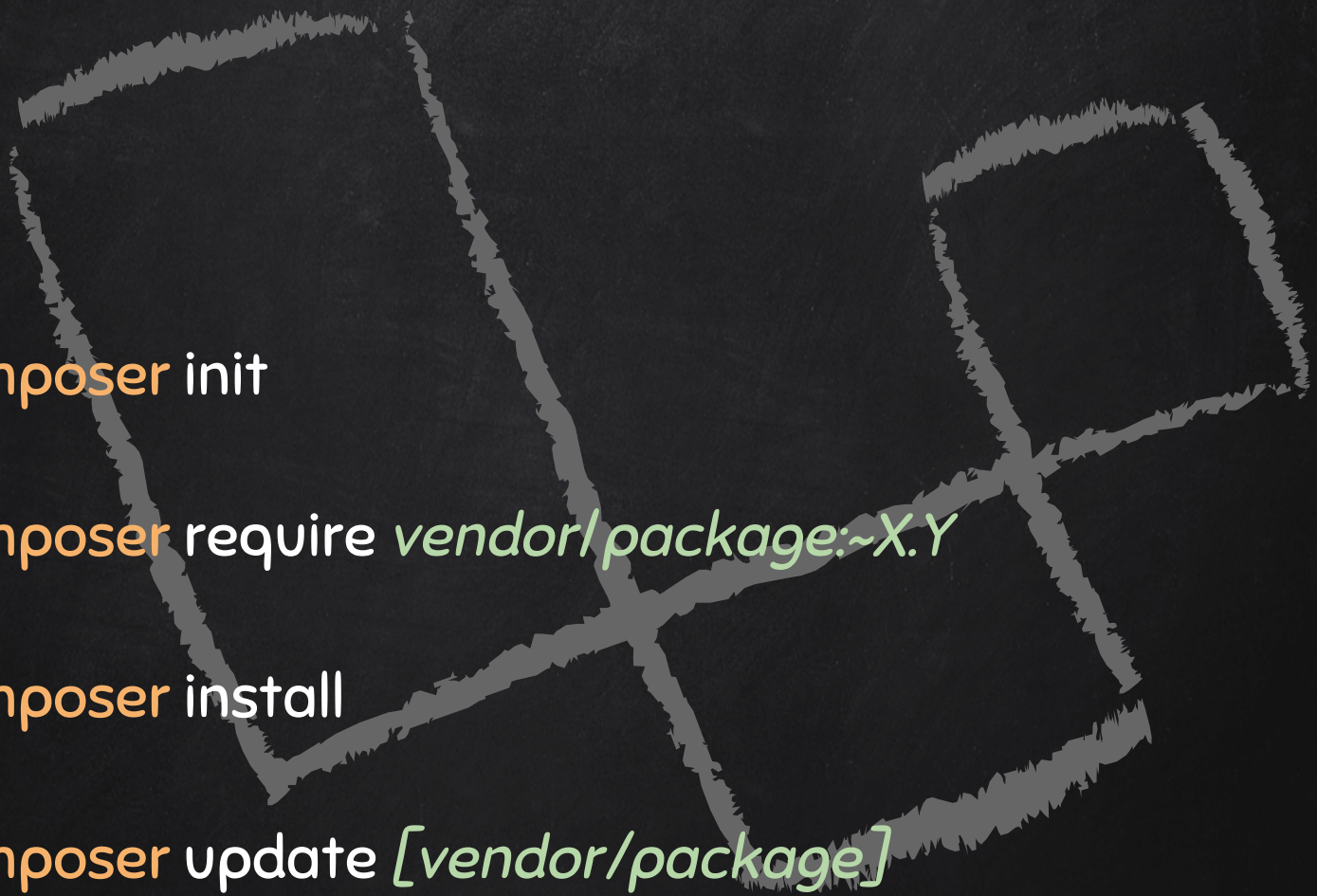
- ✘ Lenguaje de Scripting
- ✘ Tipos dinámicos
- ✘ Lenguaje interpretado
- ✘ Orientado a Objetos
- ✘ Creado para la web
- ✘ > 80% de la Internet
- ✘ Gran comunidad
- ✘ Código Abierto (*Open Source*)



Administrador de Dependencias

Composer

Composer



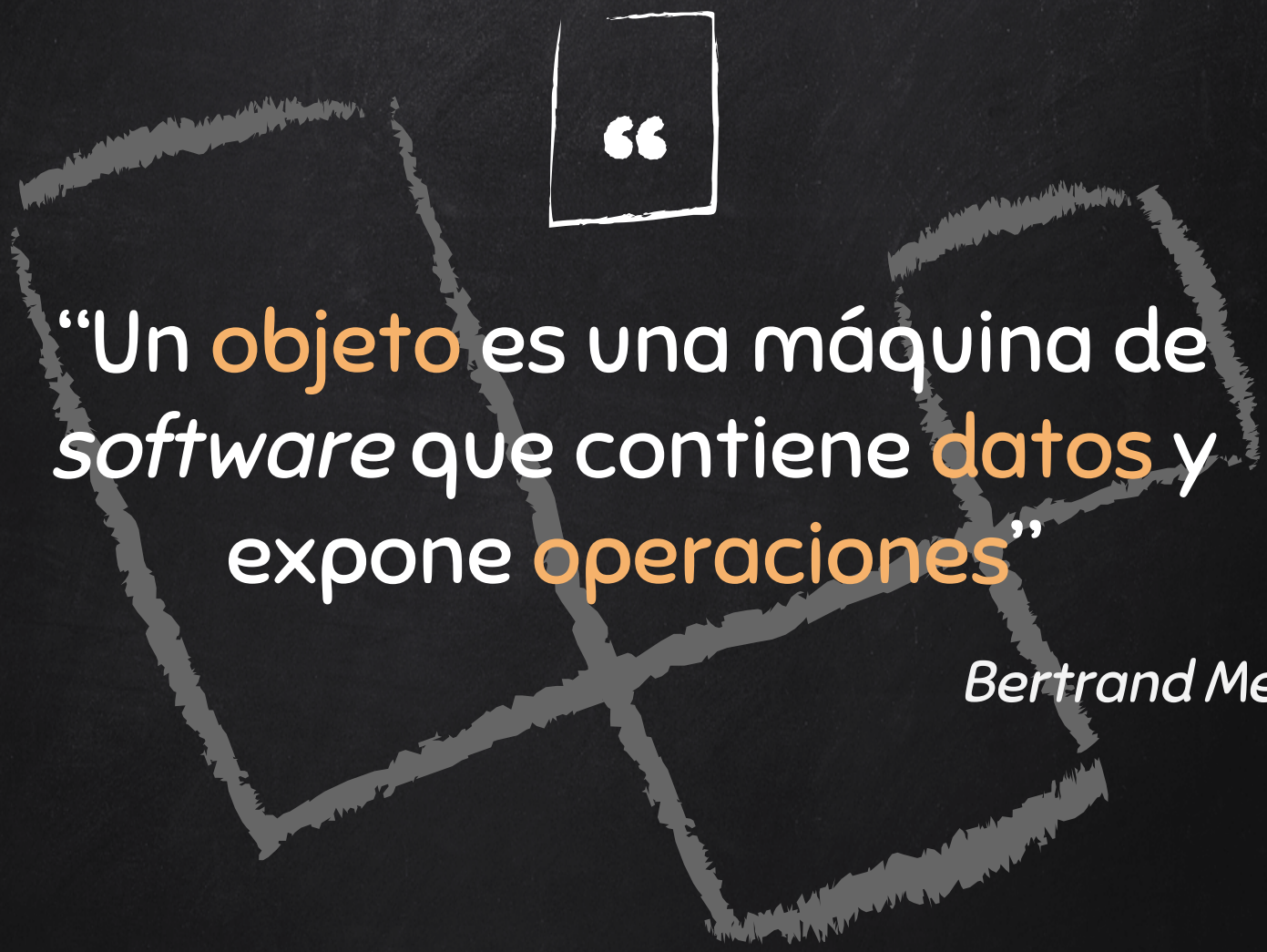
```
$ composer init

$ composer require vendor/package:~X.Y

$ composer install

$ composer update [vendor/package]
```

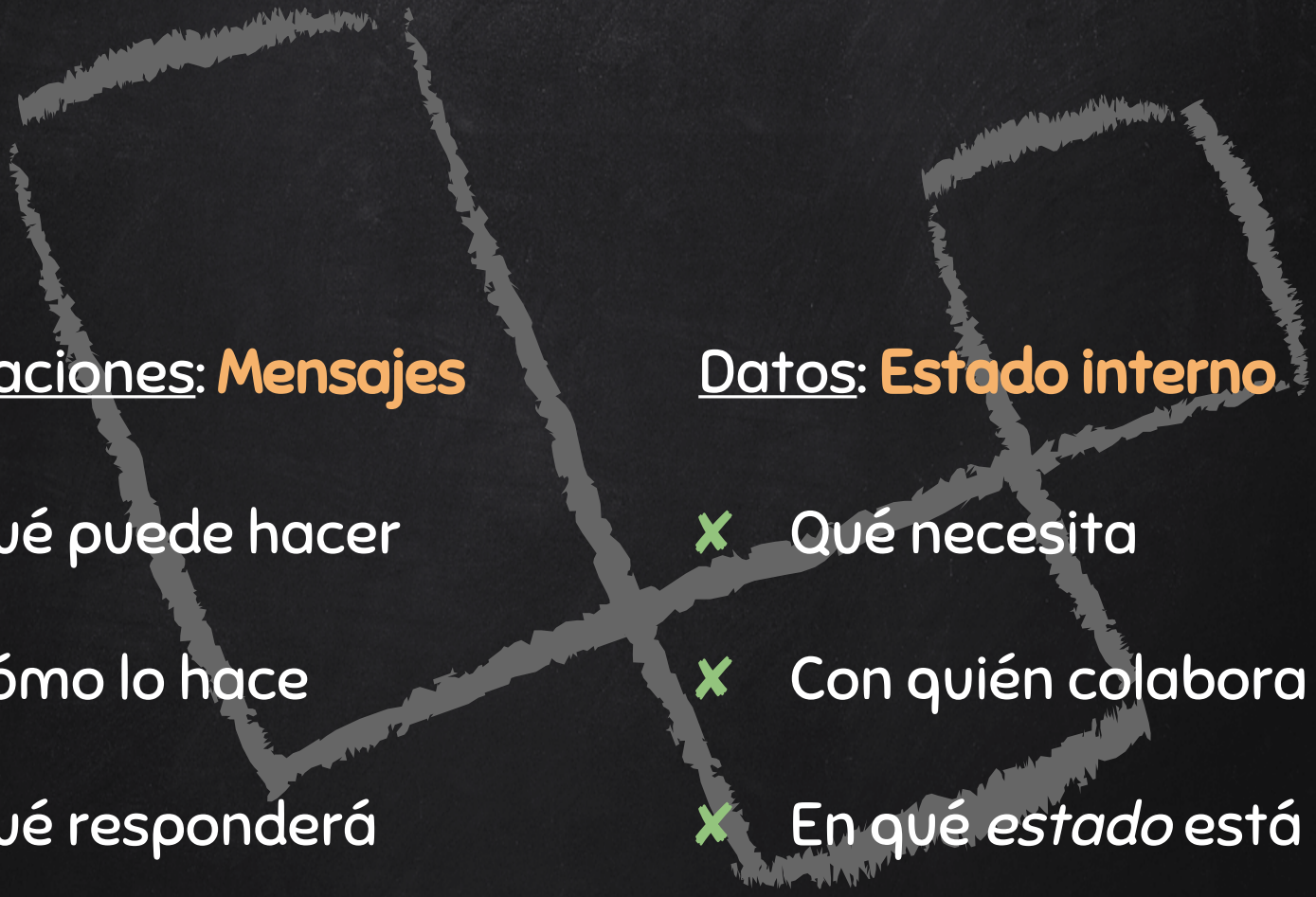

Objetos



“Un objeto es una máquina de *software* que contiene datos y expone operaciones”

Bertrand Meyer

Objetos (*Objects*)

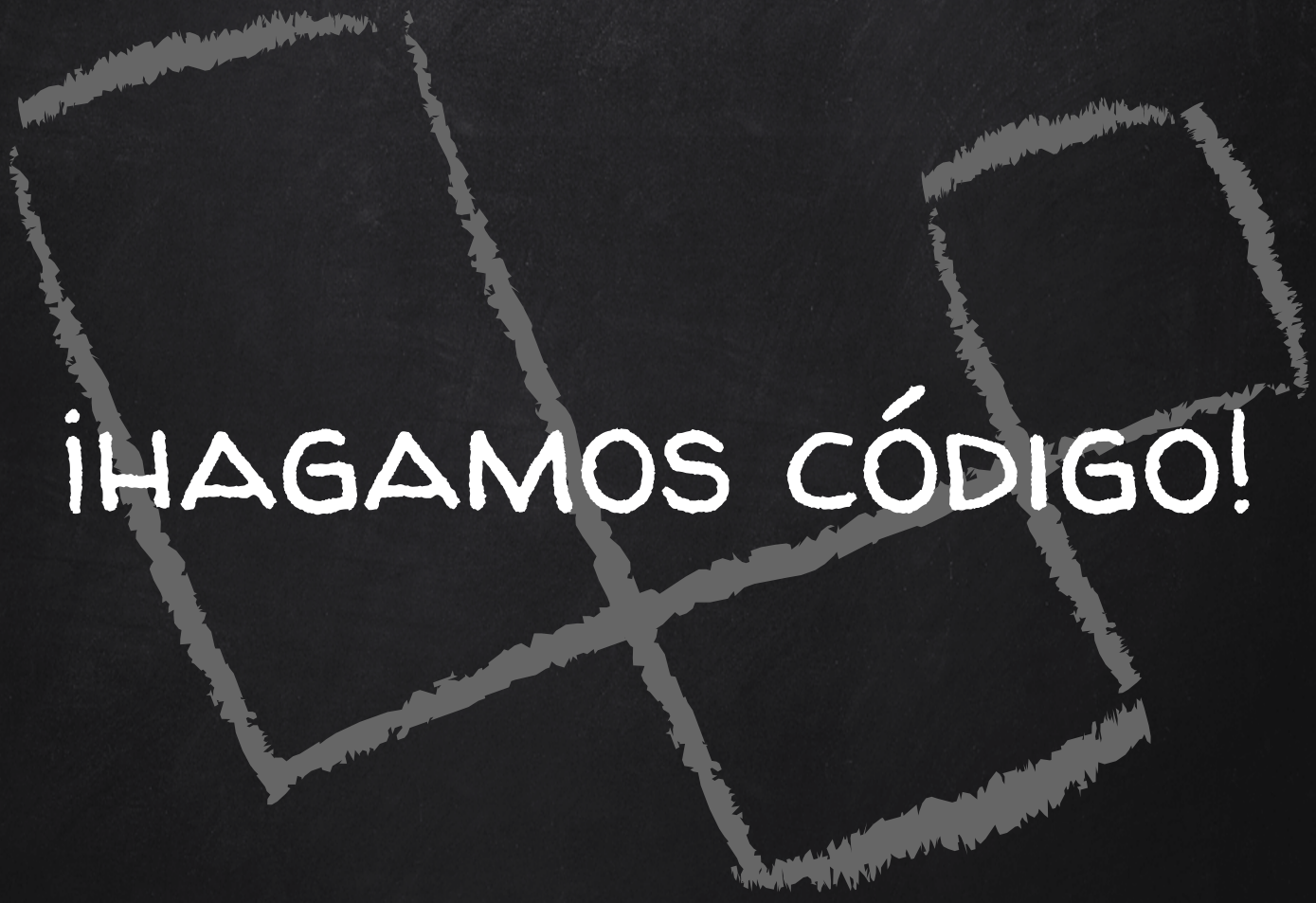


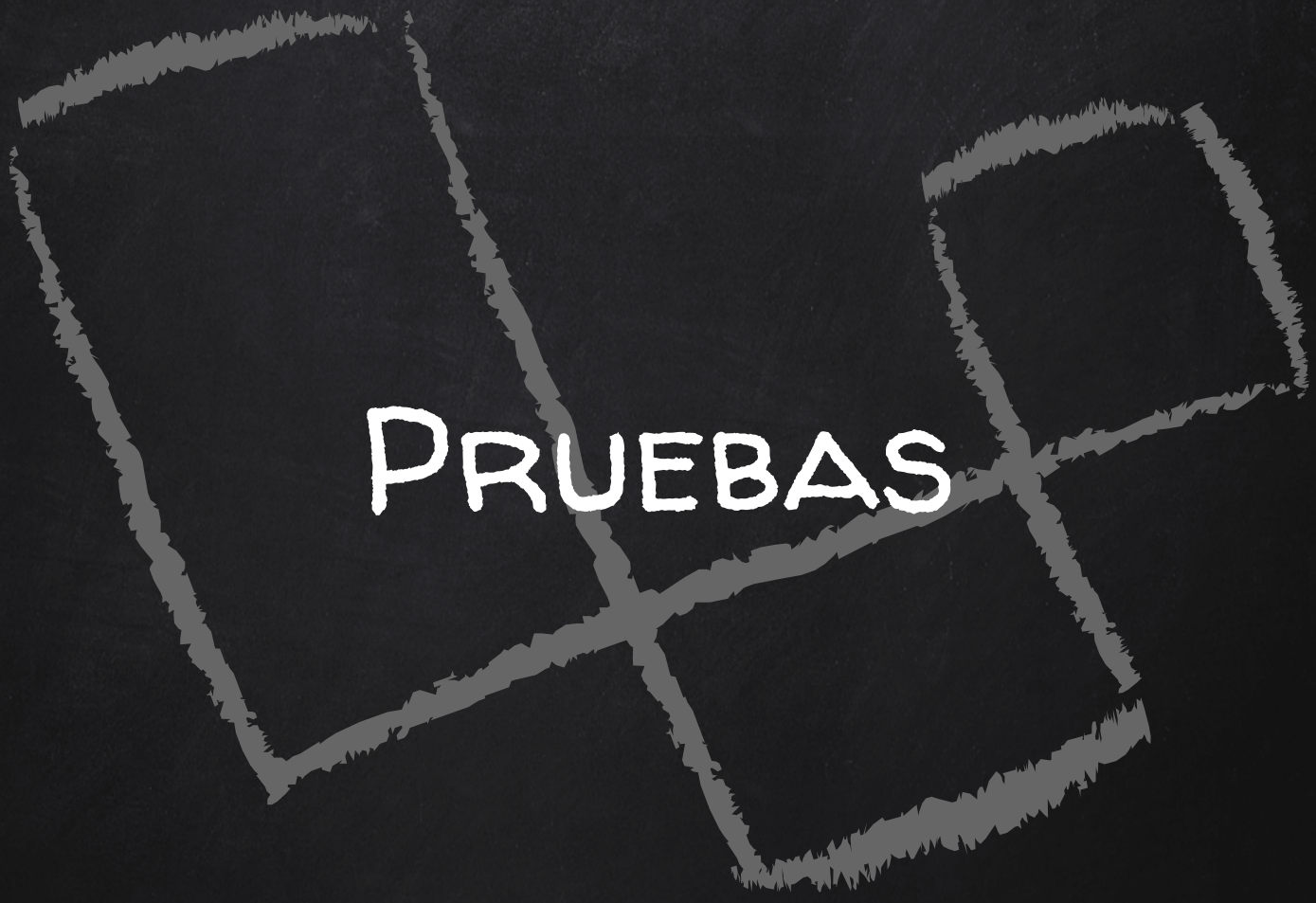
Operaciones: **Mensajes**

- ✗ Qué puede hacer
- ✗ Cómo lo hace
- ✗ Qué responderá

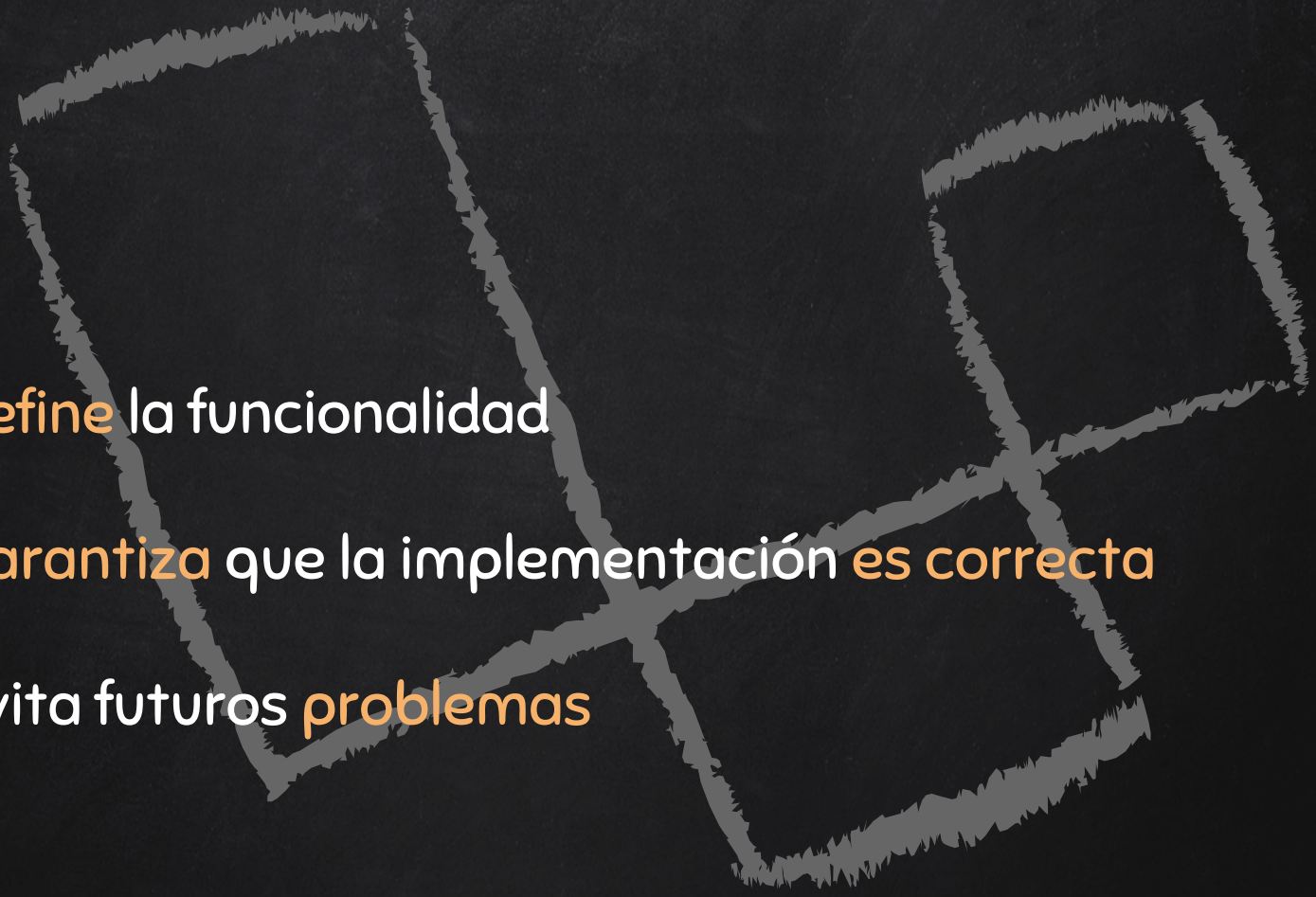
Datos: **Estado interno**

- ✗ Qué necesita
- ✗ Con quién colabora
- ✗ En qué *estado* está


¡HAGAMOS CÓDIGO!

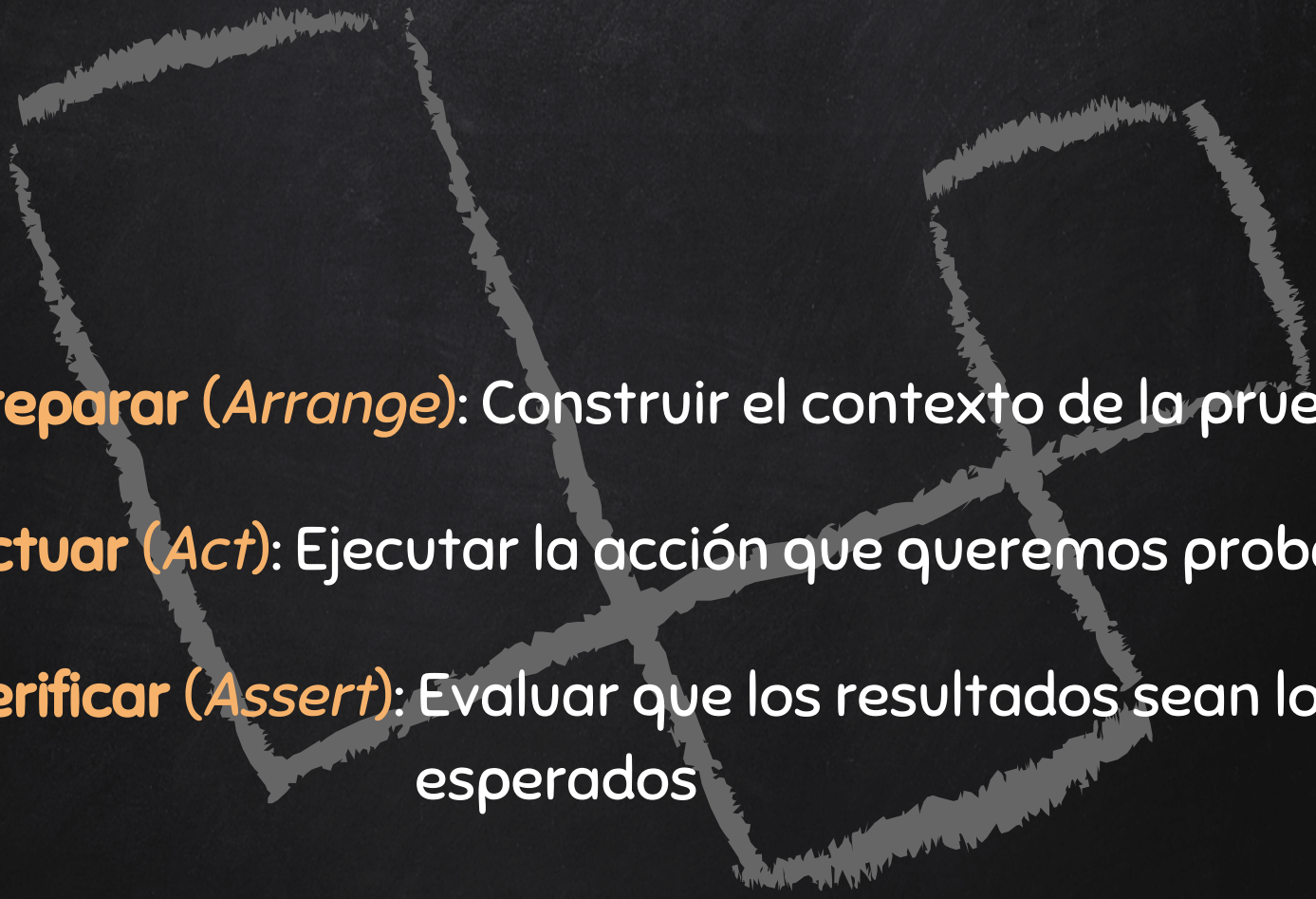

Pruebas

Pruebas (*Testing*)

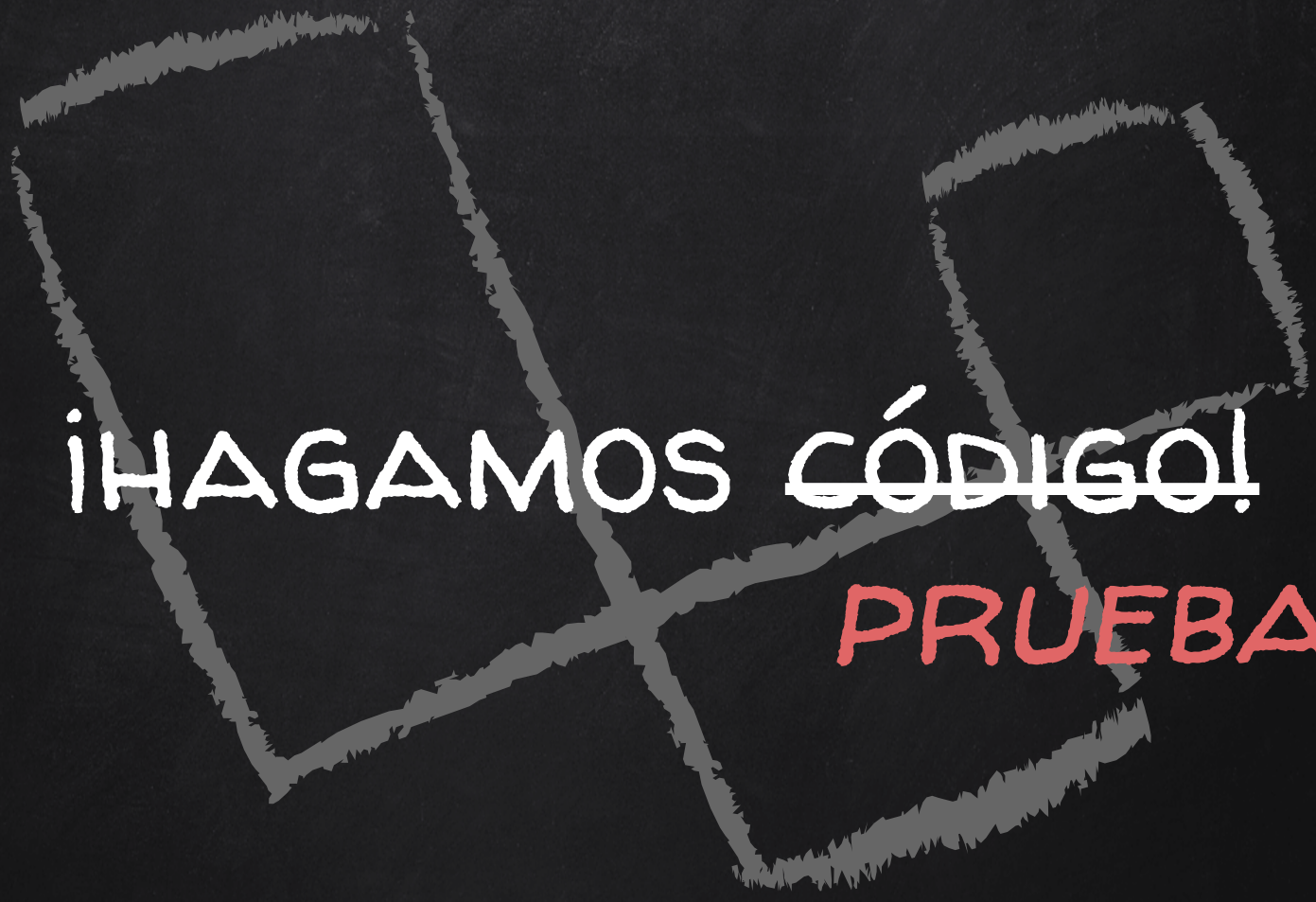


- ✘ Define la funcionalidad
- ✘ Garantiza que la implementación es correcta
- ✘ Evita futuros problemas

Pruebas (*Testing*)



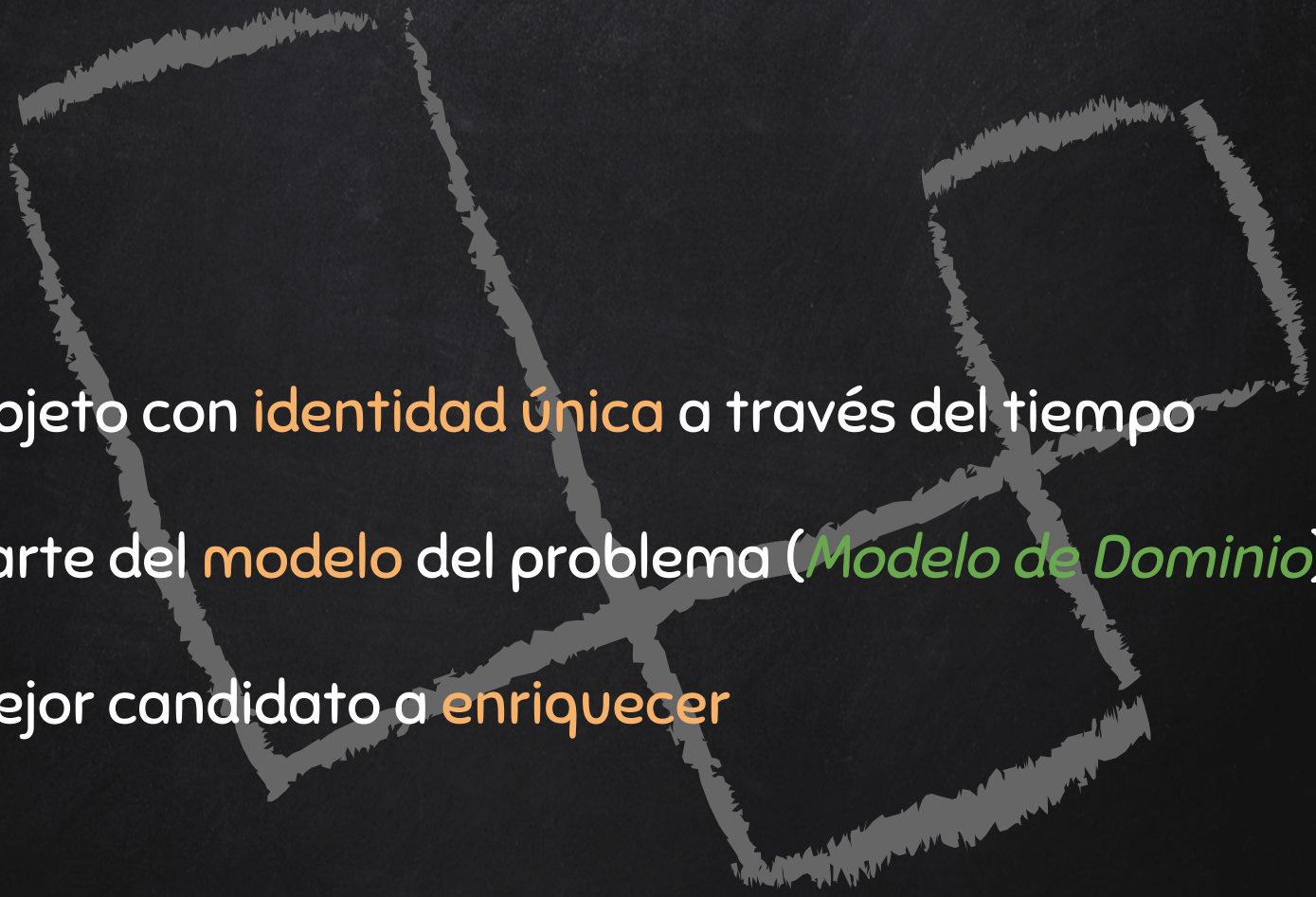
- **Preparar** (*Arrange*): Construir el contexto de la prueba
- **Actuar** (*Act*): Ejecutar la acción que queremos probar
- **Verificar** (*Assert*): Evaluar que los resultados sean los esperados


¡HAGAMOS ~~CÓDIGO~~!
PRUEBAS!

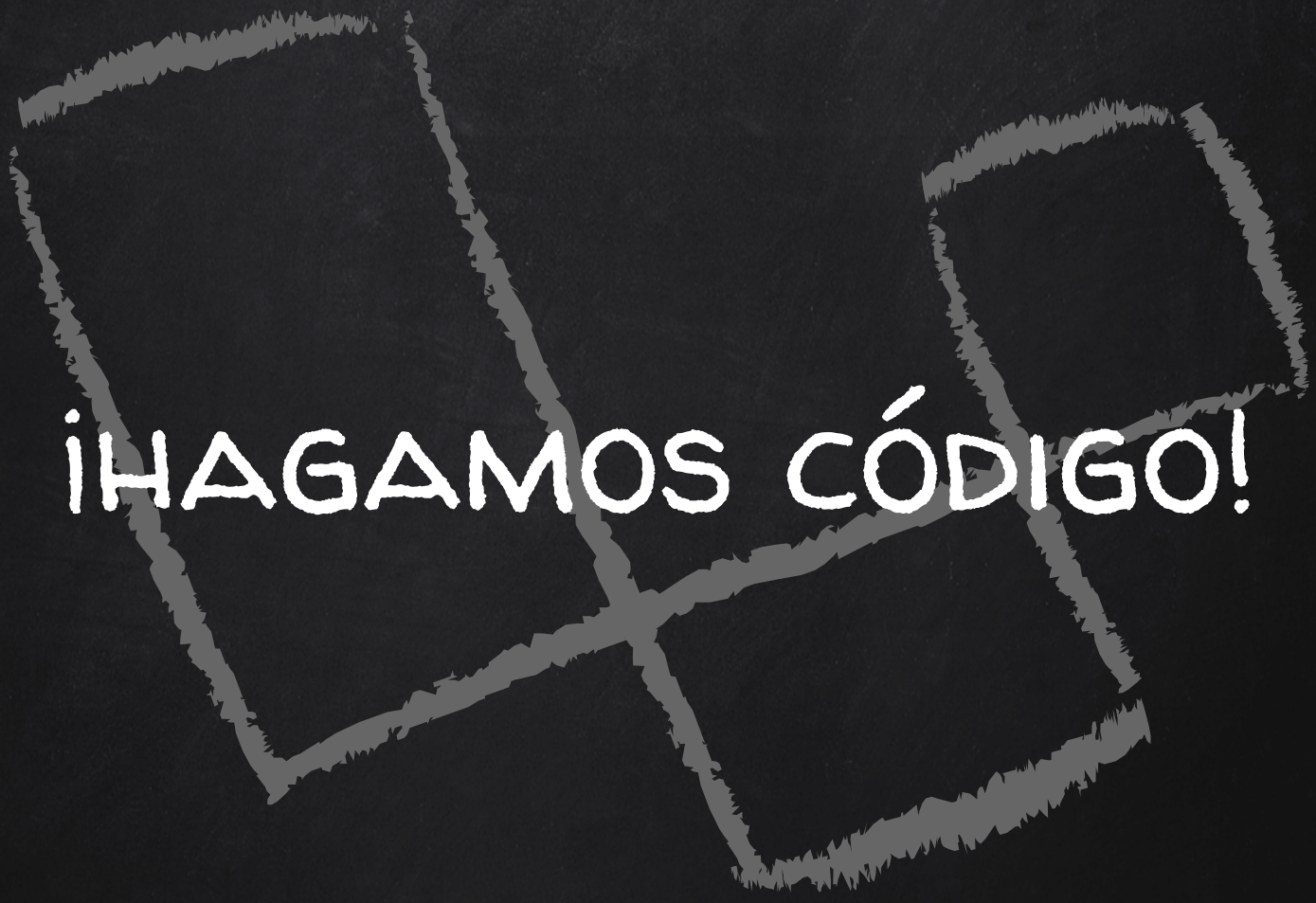

Patrones de diseño


Entidades

Entidades (*Entities*)

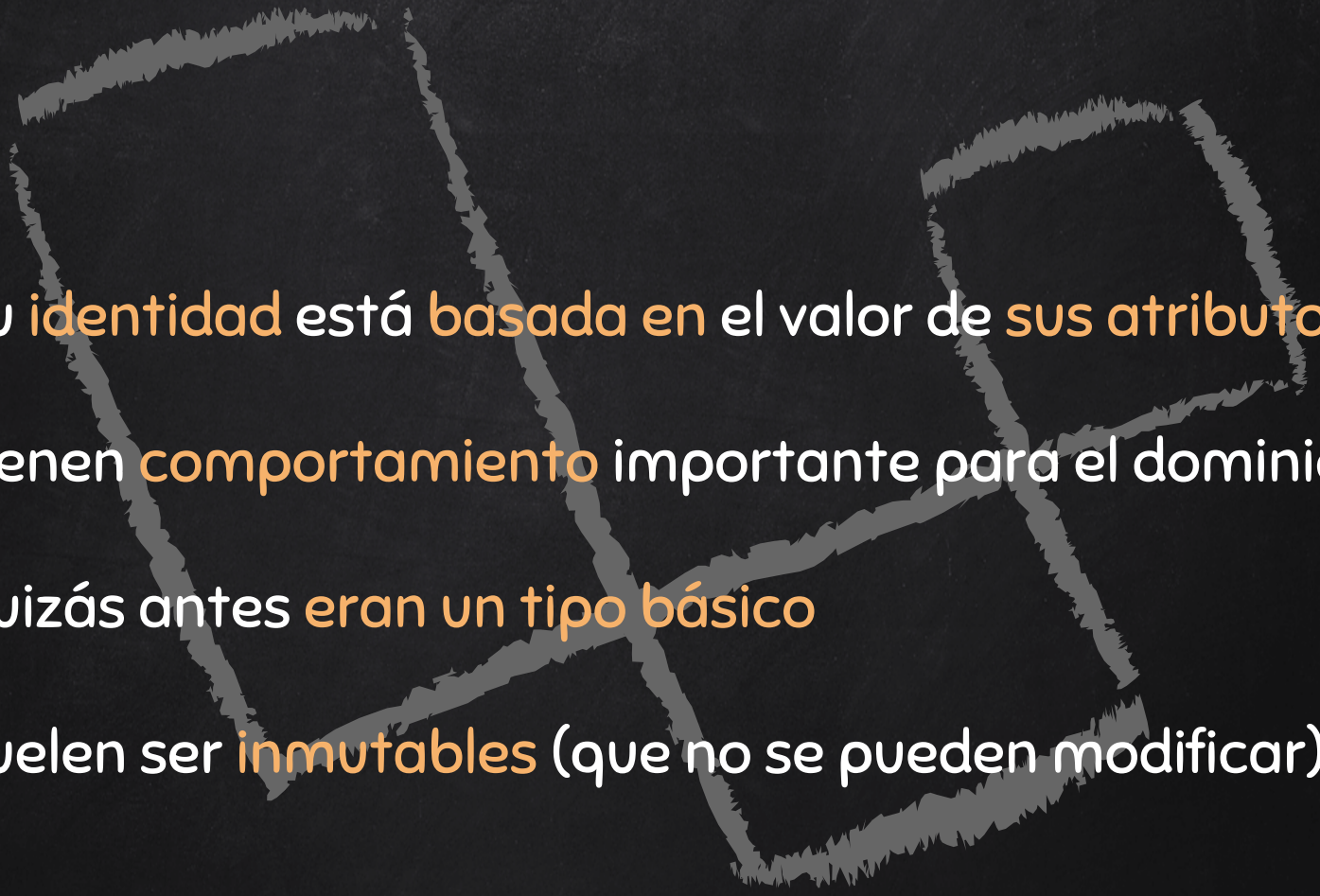


- ✘ Objeto con identidad única a través del tiempo
- ✘ Parte del modelo del problema (*Modelo de Dominio*)
- ✘ Mejor candidato a enriquecer

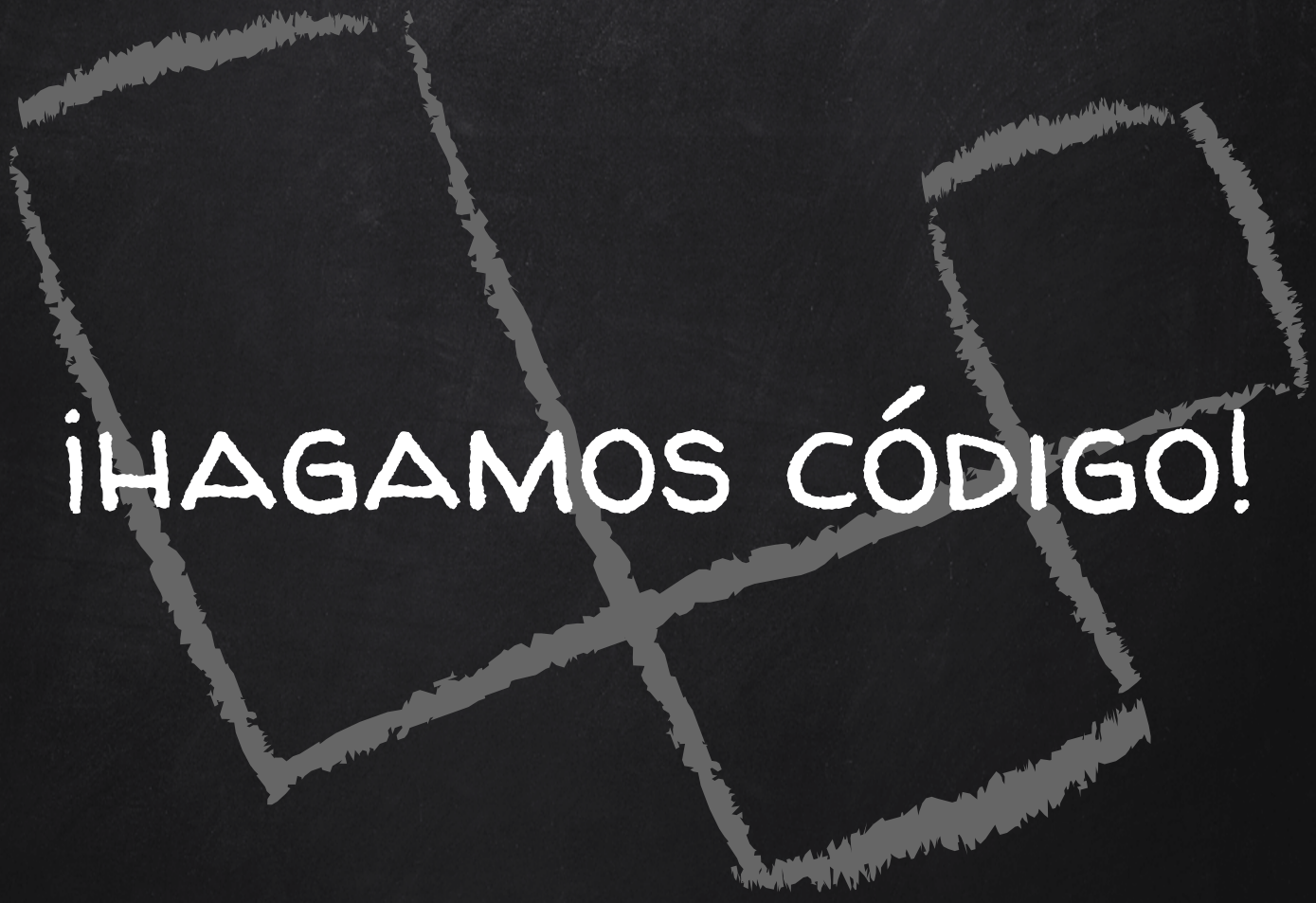

¡HAGAMOS CÓDIGO!


Objetos de Valor

Objetos de valor (*Value Objects*)



- Su identidad está basada en el valor de sus atributos
- Tienen comportamiento importante para el dominio
- Quizás antes eran un tipo básico
- Suelen ser inmutables (que no se pueden modificar)


¡HAGAMOS CÓDIGO!


Resumen

Resumen

Objetos

- ✘ Estado
- ✘ Mensajes
- ✘ Colaboración
- ✘ Herencia
- ✘ Responsabilidad

Composer

- ✘ Dependencias
- ✘ Autoloading

Testing

- ✘ Definición
- ✘ Comprobación
- ✘ Mantenimiento

Entidades

- ✘ Identidad
- ✘ Modelo del problema

Objetos de Valor

- ✘ Id <=> estado
- ✘ Inmutables

¡Hasta la próxima!

PHP y Laravel

Clase 2

PHP en la web

Guido Contreras Woda - @guiwoda

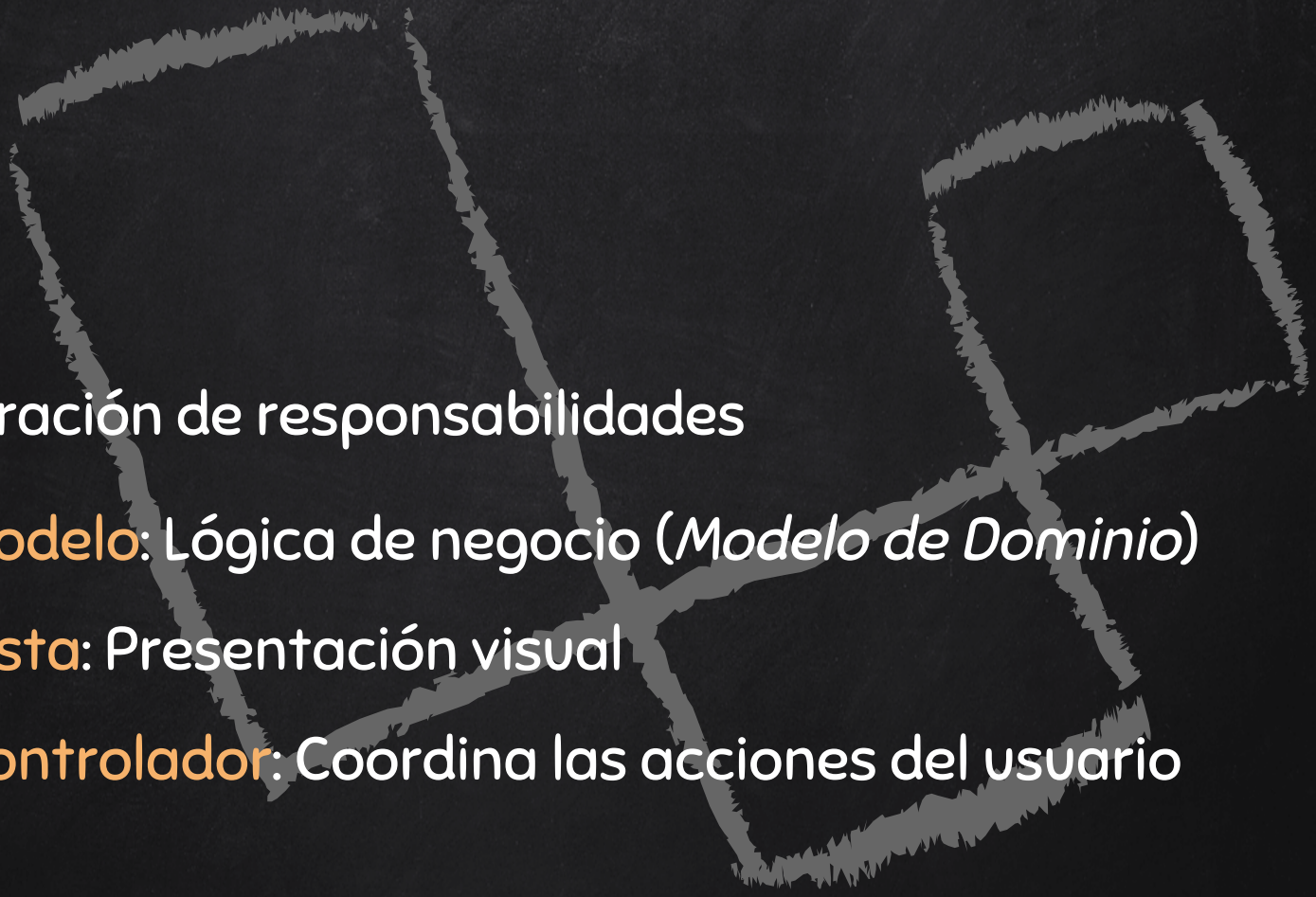
¿Qué es la web?

- ✗ Protocolo HTTP
- ✗ Lenguaje HTML
- ✗ Arquitectura Cliente – Servidor
- ✗ Antes: Pedir un archivo
- ✗ PHP Hoy: Ruteo de *URI*
- ✗ PHP + JS: JSON *API* + *Client-side programming*

Patrones de Arquitectura

Modelo Vista Controlador (MVC)

Modelo Vista Controlador (*MVC*)

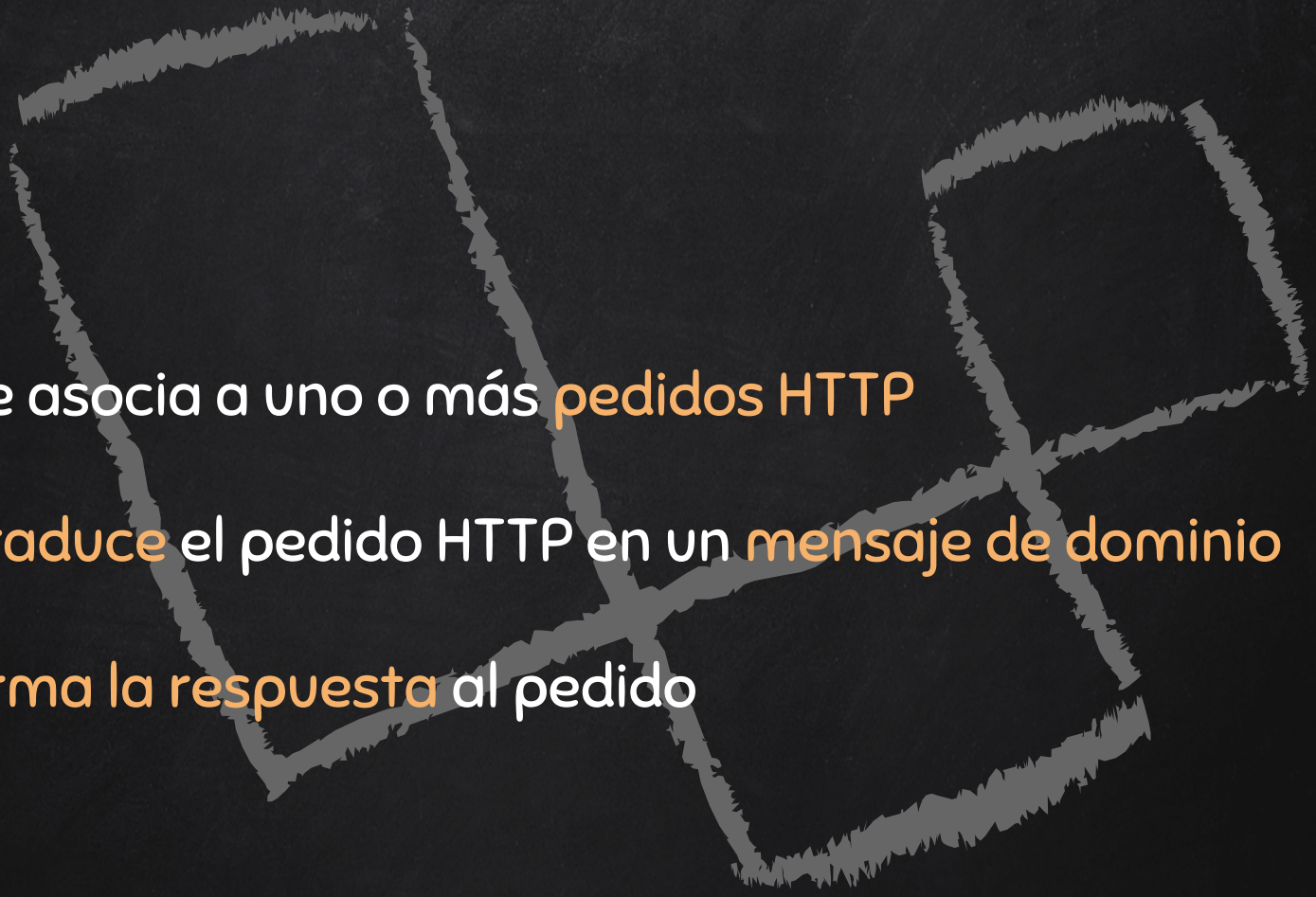


Separación de responsabilidades

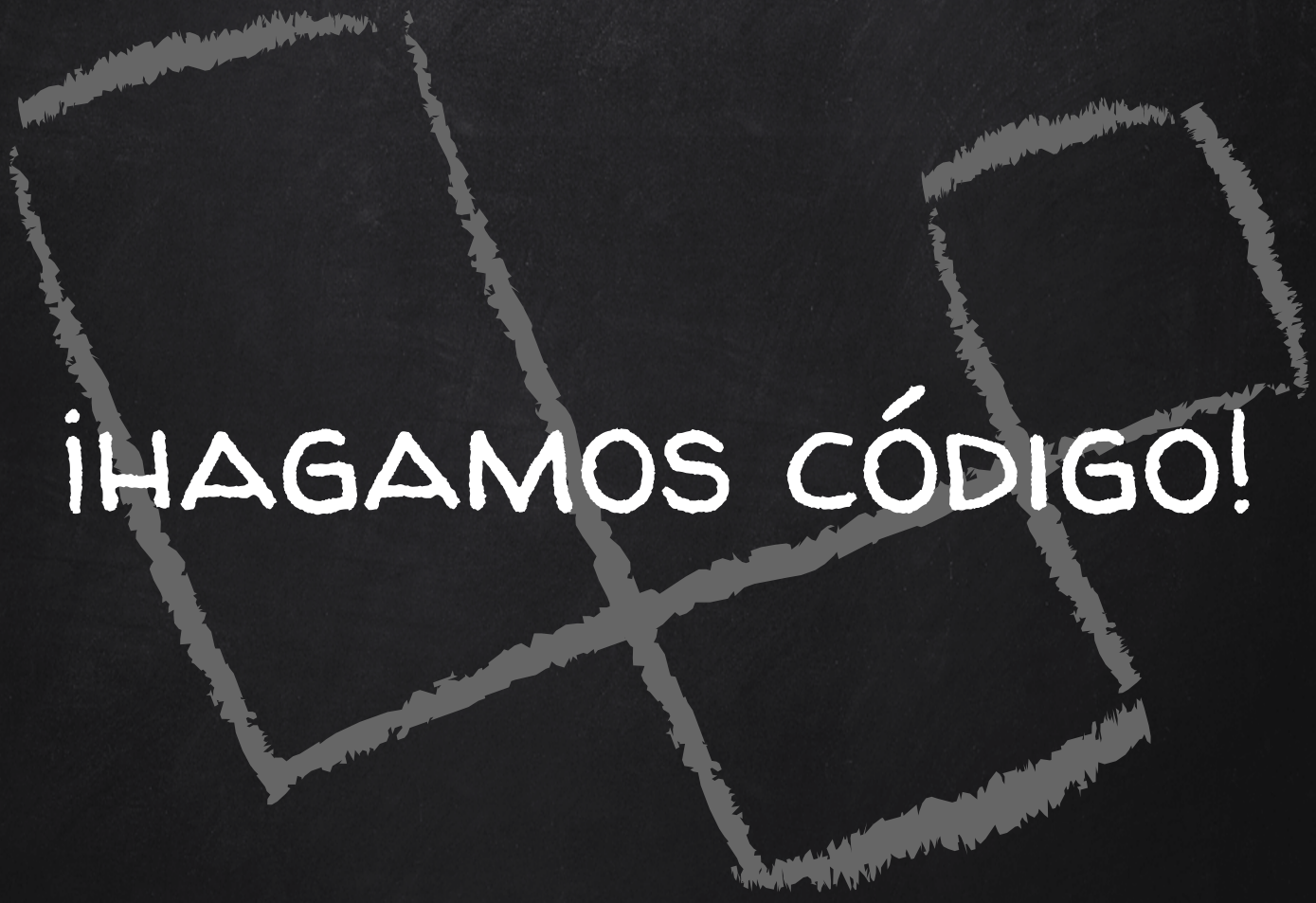
- Modelo: Lógica de negocio (*Modelo de Dominio*)
- Vista: Presentación visual
- Controlador: Coordina las acciones del usuario


Controlador

Controlador (Controller)

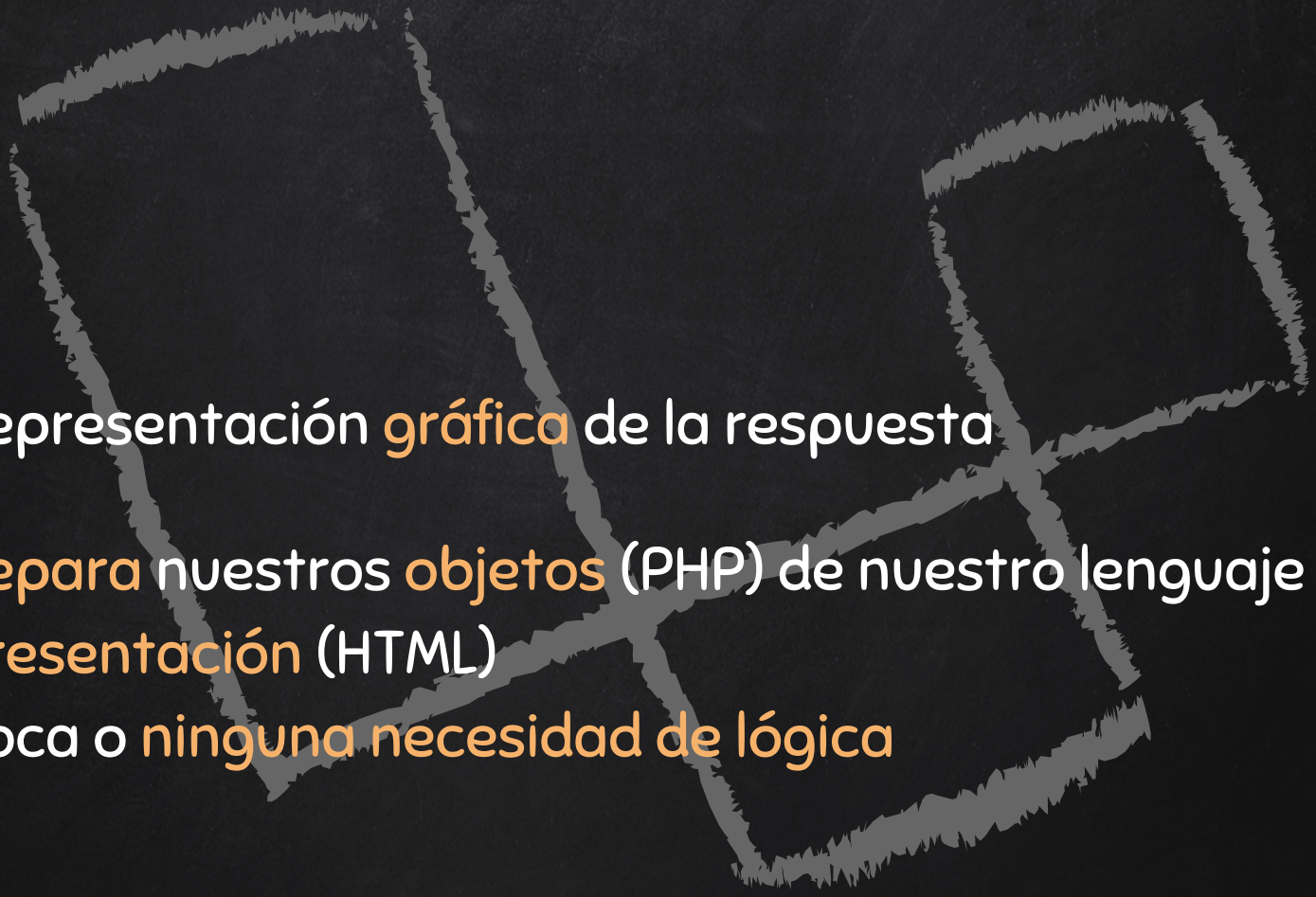


- Se asocia a uno o más pedidos HTTP
- Traduce el pedido HTTP en un mensaje de dominio
- Arma la respuesta al pedido

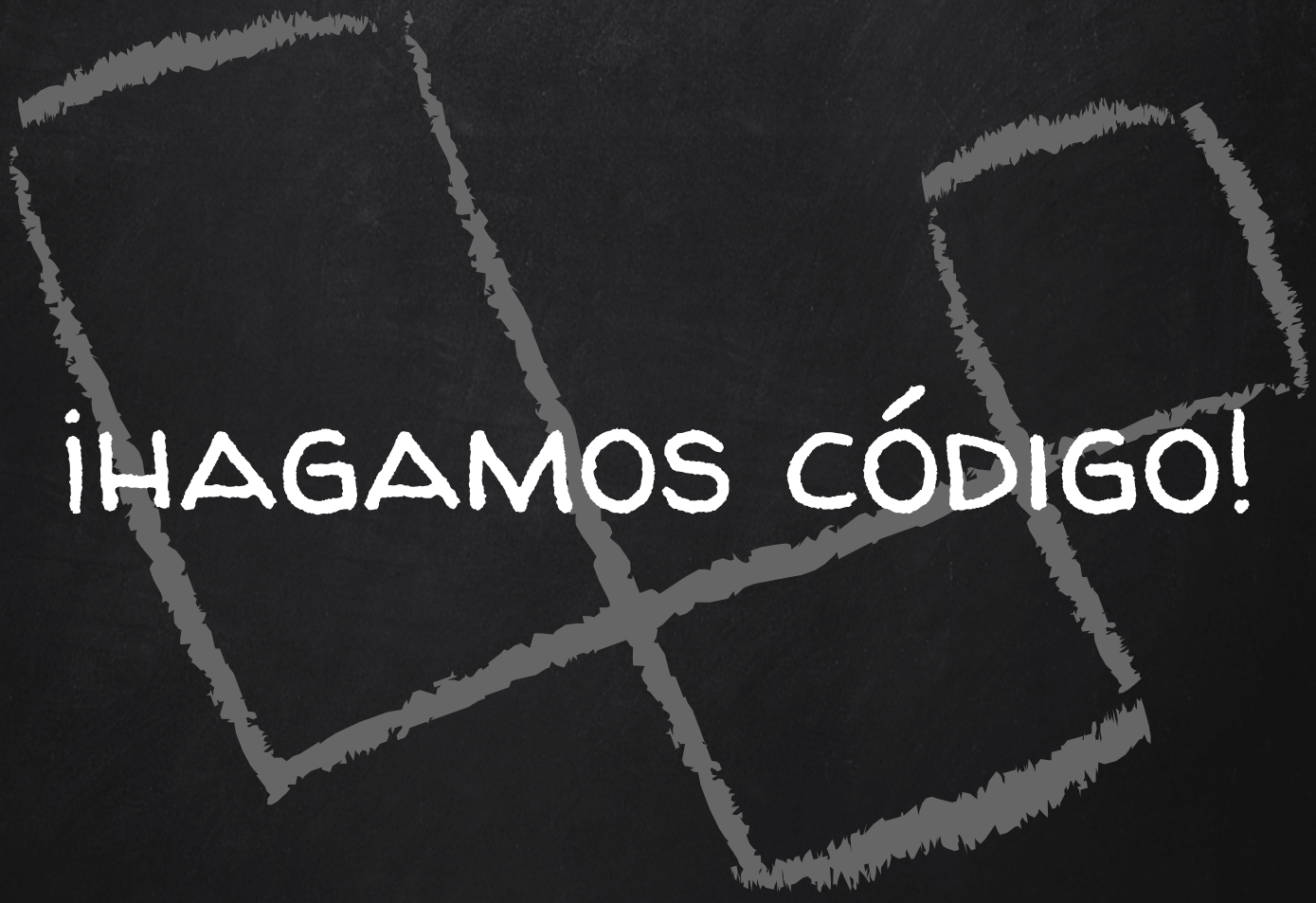

¡HAGAMOS CÓDIGO!

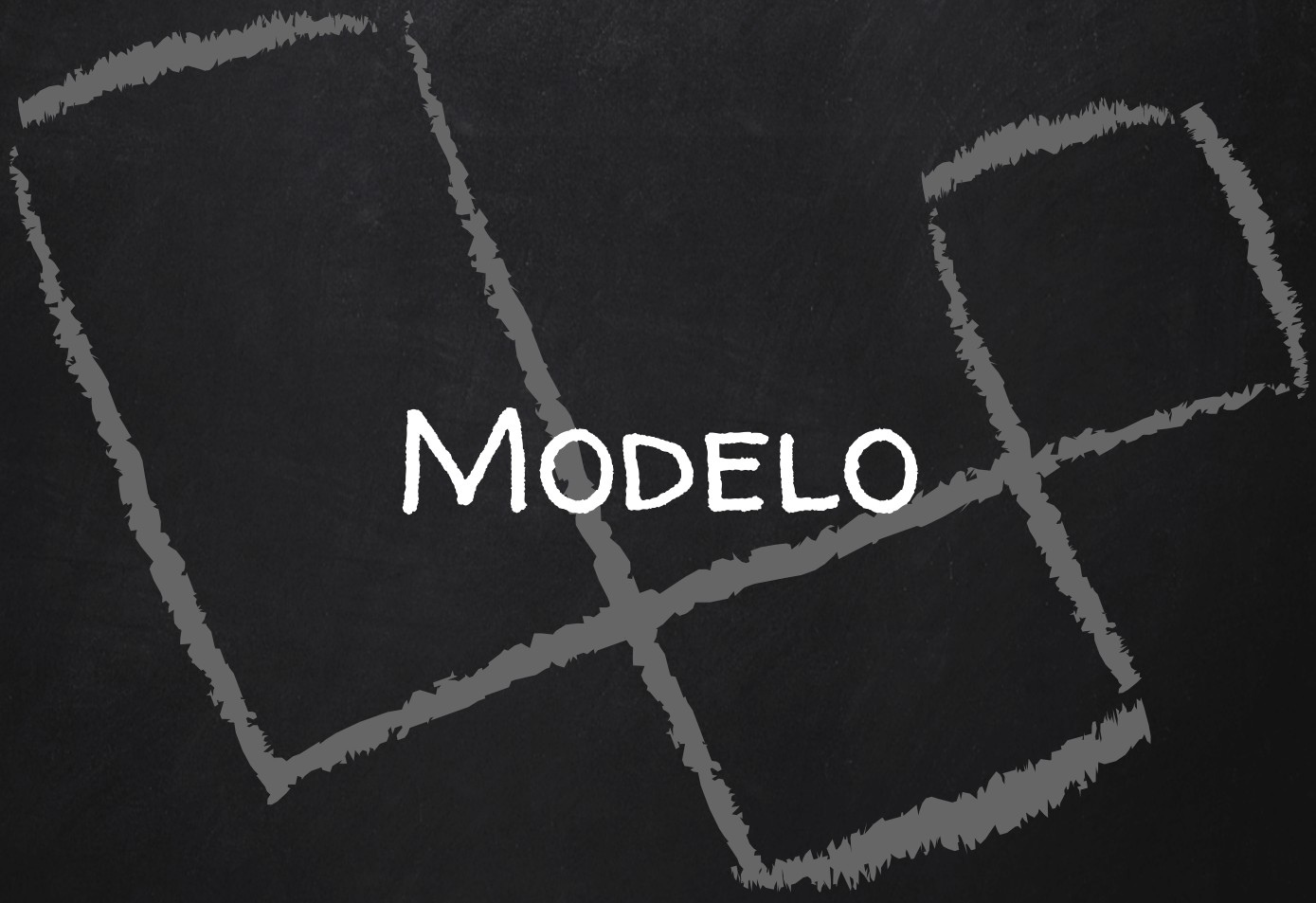

Vista

Vista (*View*)

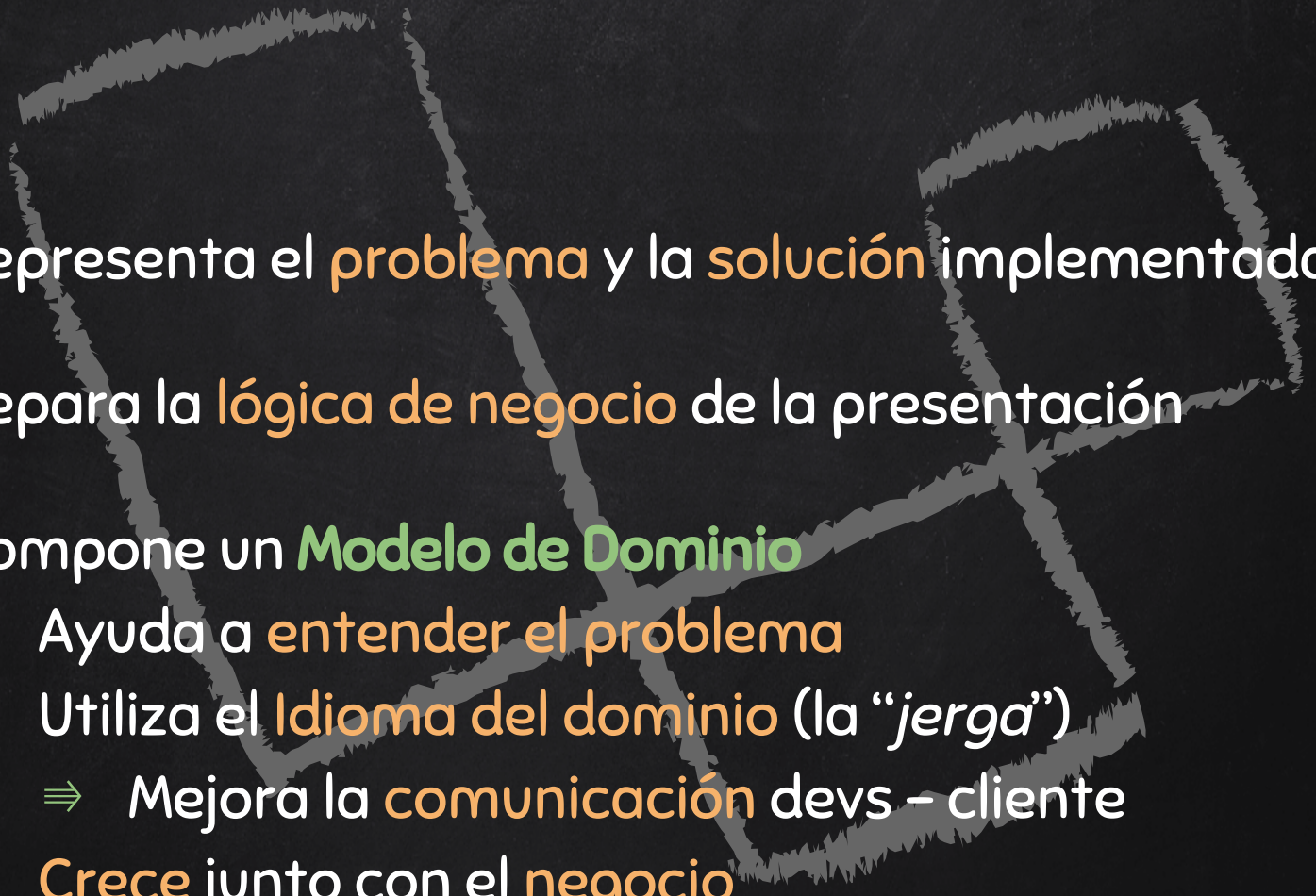


- ✘ Representación gráfica de la respuesta
- ✘ Separa nuestros objetos (PHP) de nuestro lenguaje de presentación (HTML)
- ✘ Poca o ninguna necesidad de lógica

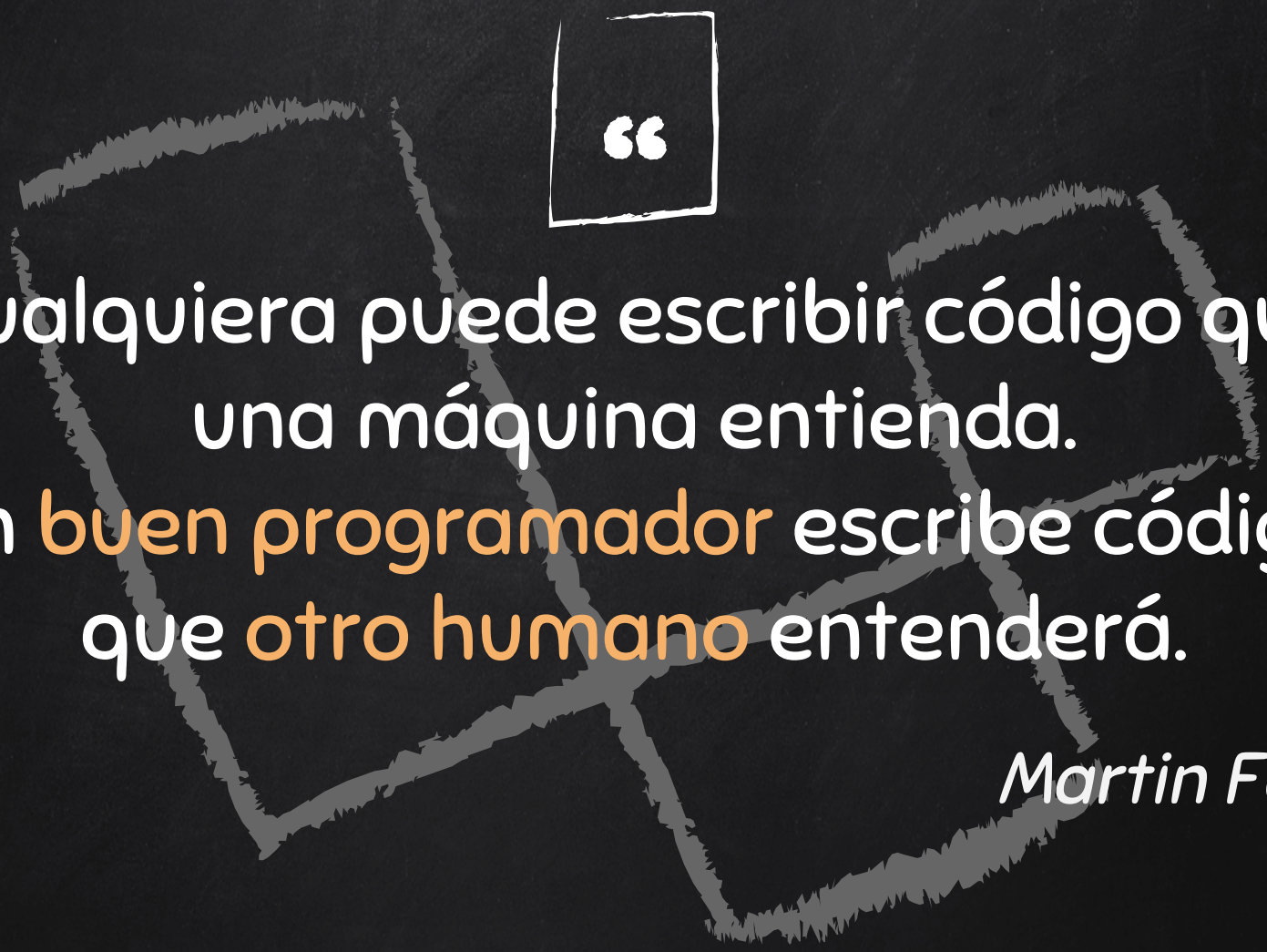

¡HAGAMOS CÓDIGO!


Modelo

Modelo (*Model*)

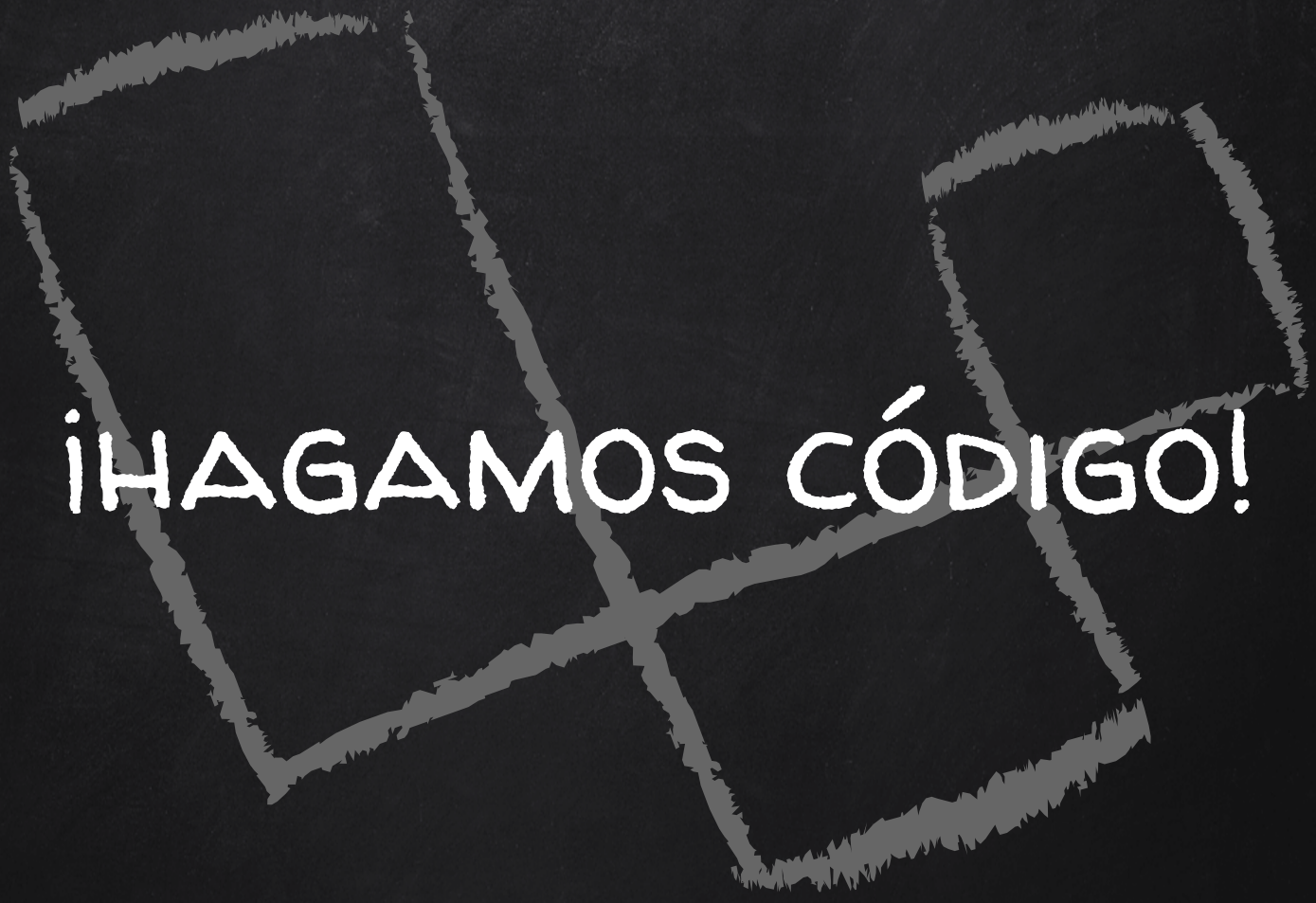


- ✘ Representa el problema y la solución implementada
- ✘ Separa la lógica de negocio de la presentación
- ✘ Compone un **Modelo de Dominio**
  - ✓ Ayuda a entender el problema
  - ✓ Utiliza el Idioma del dominio (la "*jerga*")
    - ⇛ Mejora la comunicación devs – cliente
  - ✓ Crece junto con el negocio



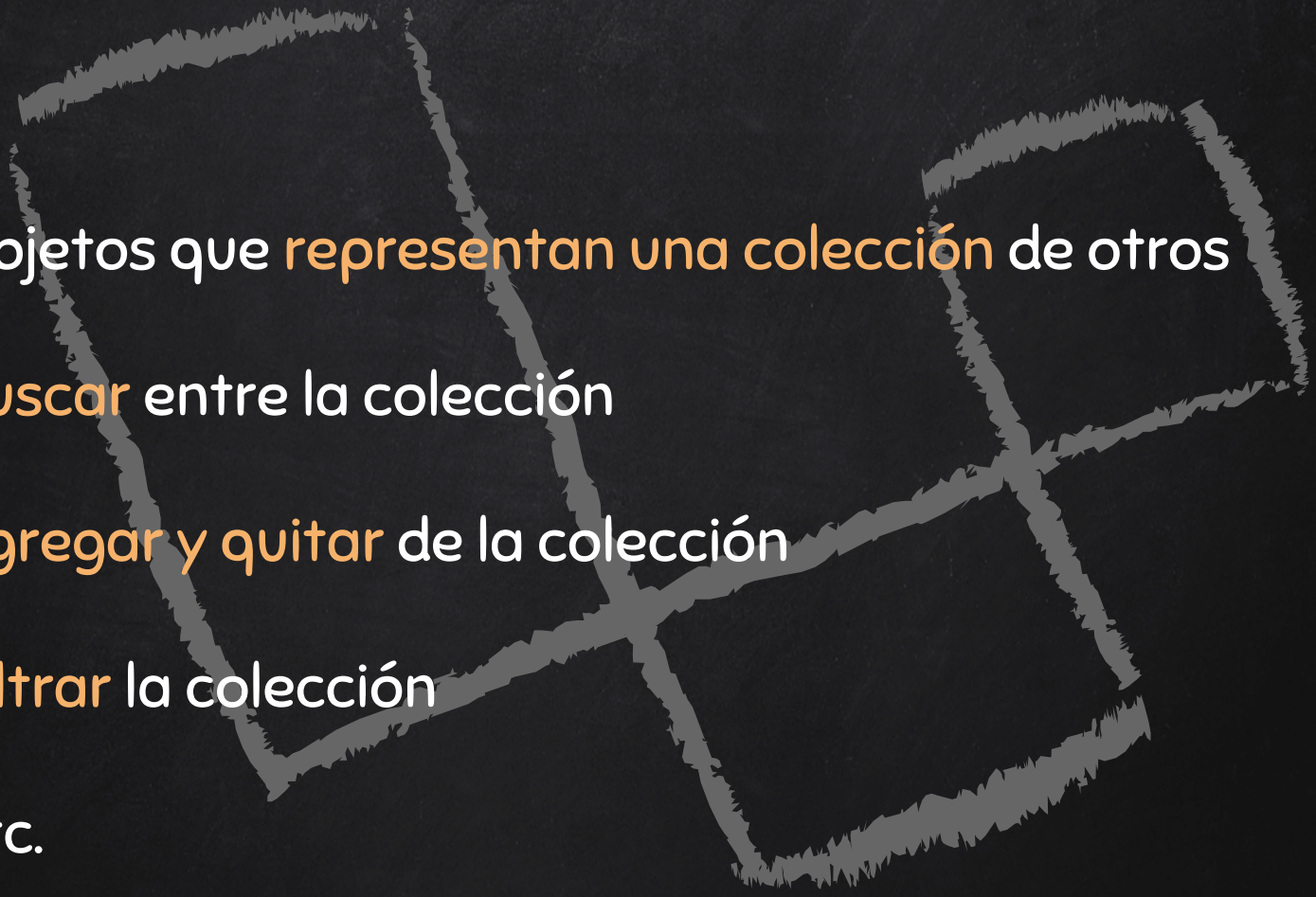
Cualquiera puede escribir código que una máquina entienda.
Un buen programador escribe código que otro humano entenderá.

Martin Fowler

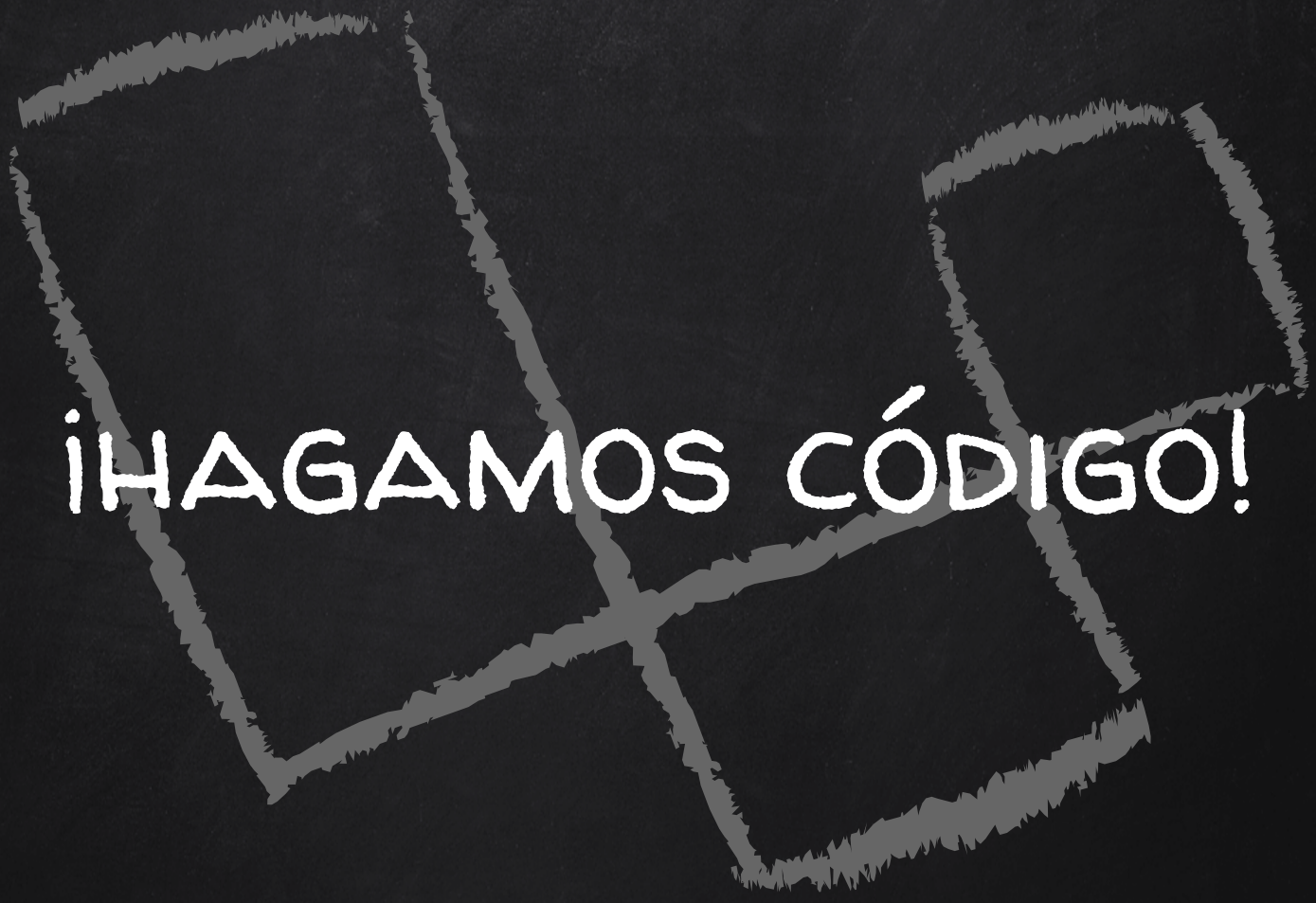

¡HAGAMOS CÓDIGO!


Colecciones

Colecciones (*Collections*)



- Objetos que representan una colección de otros
- Buscar entre la colección
- Agregar y quitar de la colección
- Filtrar la colección
- etc.


¡HAGAMOS CÓDIGO!


Resumen

Resumen

Web

- ✗ Protocolo HTTP
- ✗ HTML (CSS – JS)
- ✗ Cliente – Servidor

MVC

- ✗ Patrón de Arq.
- ✗ Separación de Responsabilidad

Controladores

- ✗ Reciben pedido
- ✗ Delegan mensaje
- ✗ Arman la vista

Vista

- ✗ Representa la gráfica
- ✗ Poca lógica

Modelo

- ✗ Problema
- ✗ Comunicación
- ✗ Idioma, jerga

Colecciones

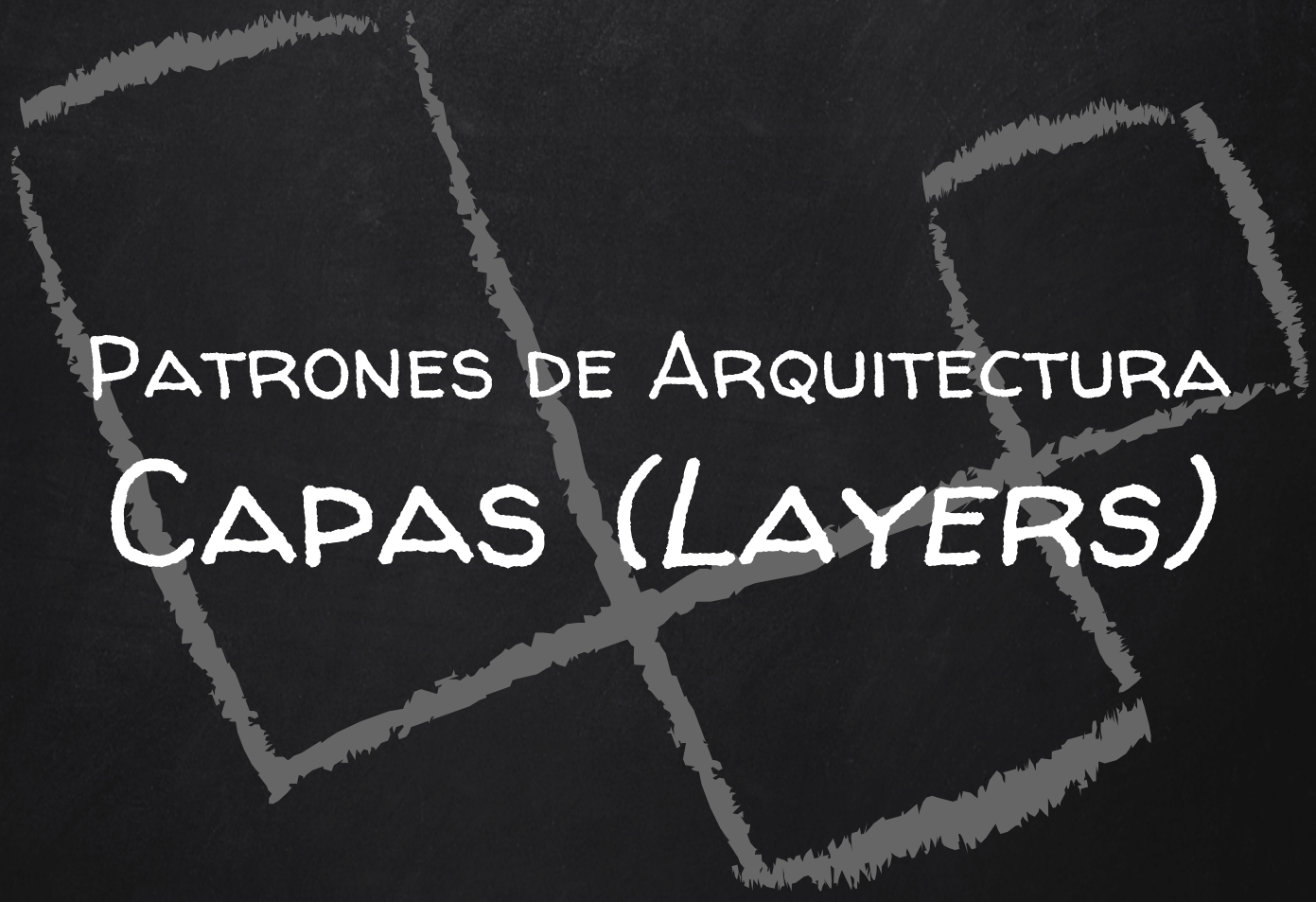
- ✗ Contiene objs.
- ✗ Mensajes más claros

¡Hasta la próxima!

PHP y Laravel

Clase 3
Buenas Prácticas

Guido Contreras Woda - @guiwoda



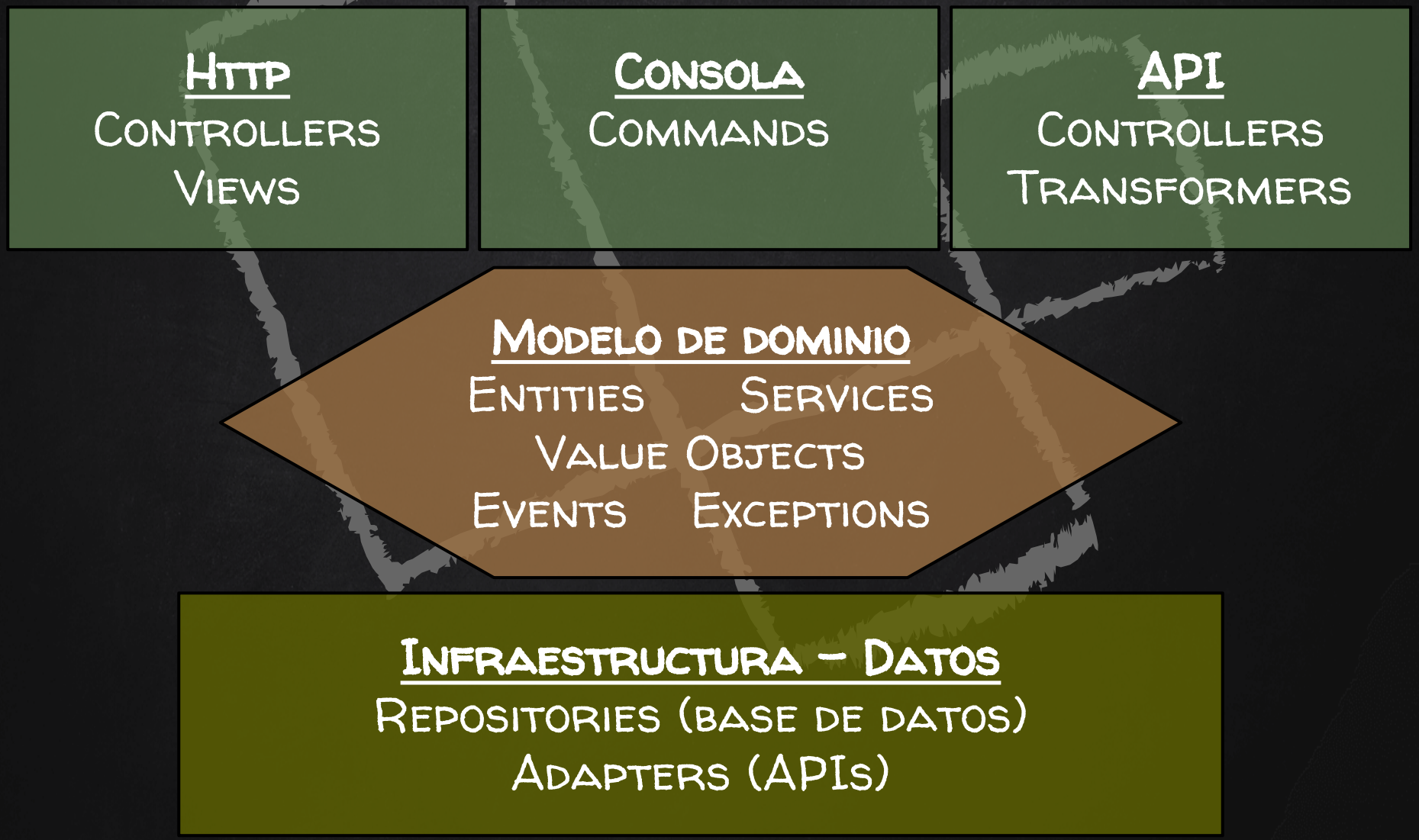
Patrones de Arquitectura

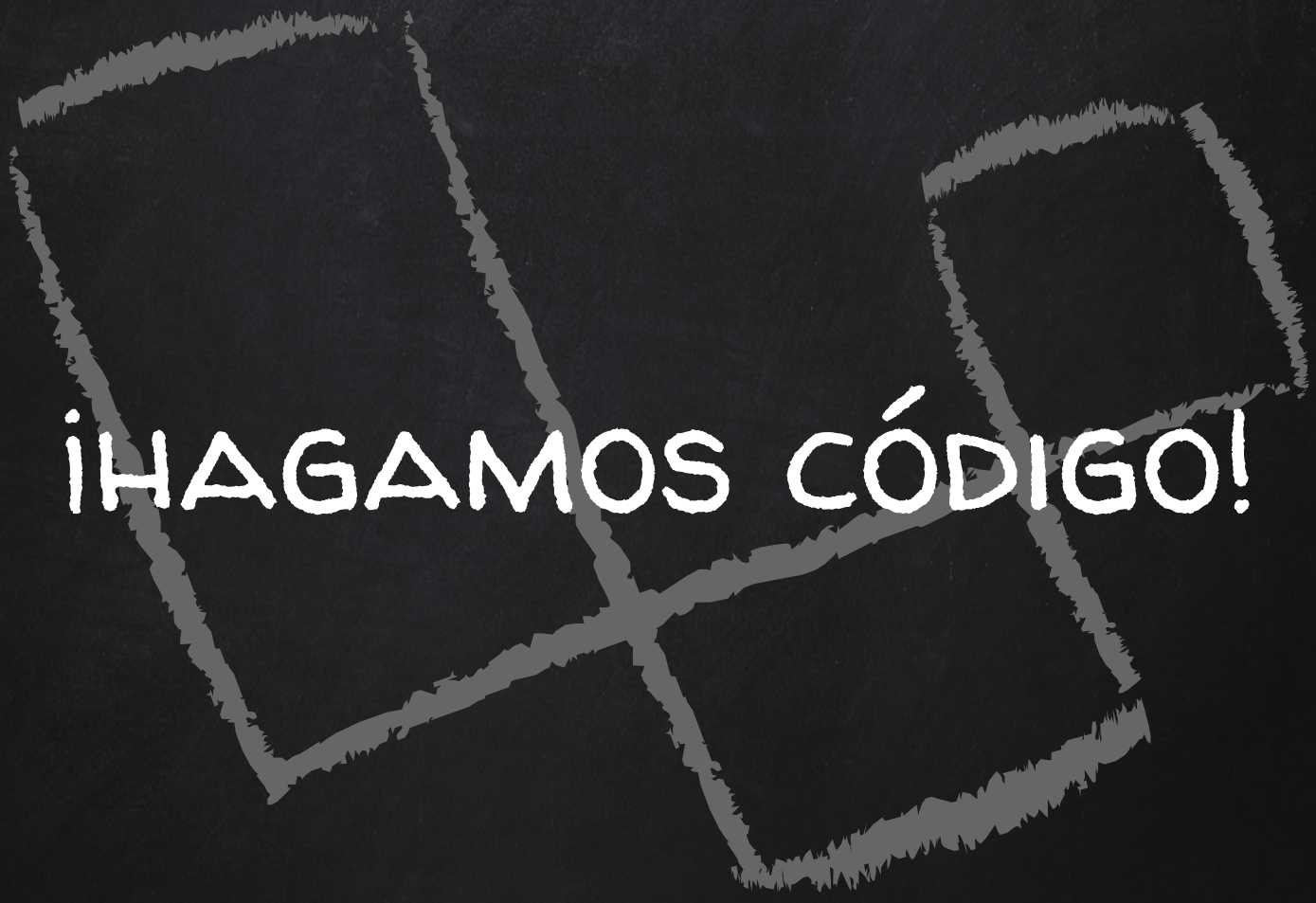
Capas (Layers)

Capas (*Layers*)

- ✘ Separa los roles a nivel de arquitectura
- ✘ Define la interacción entre cada rol
- ✘ Ayuda a construir software más robusto
- ✘ Evita el código *spaghetti*

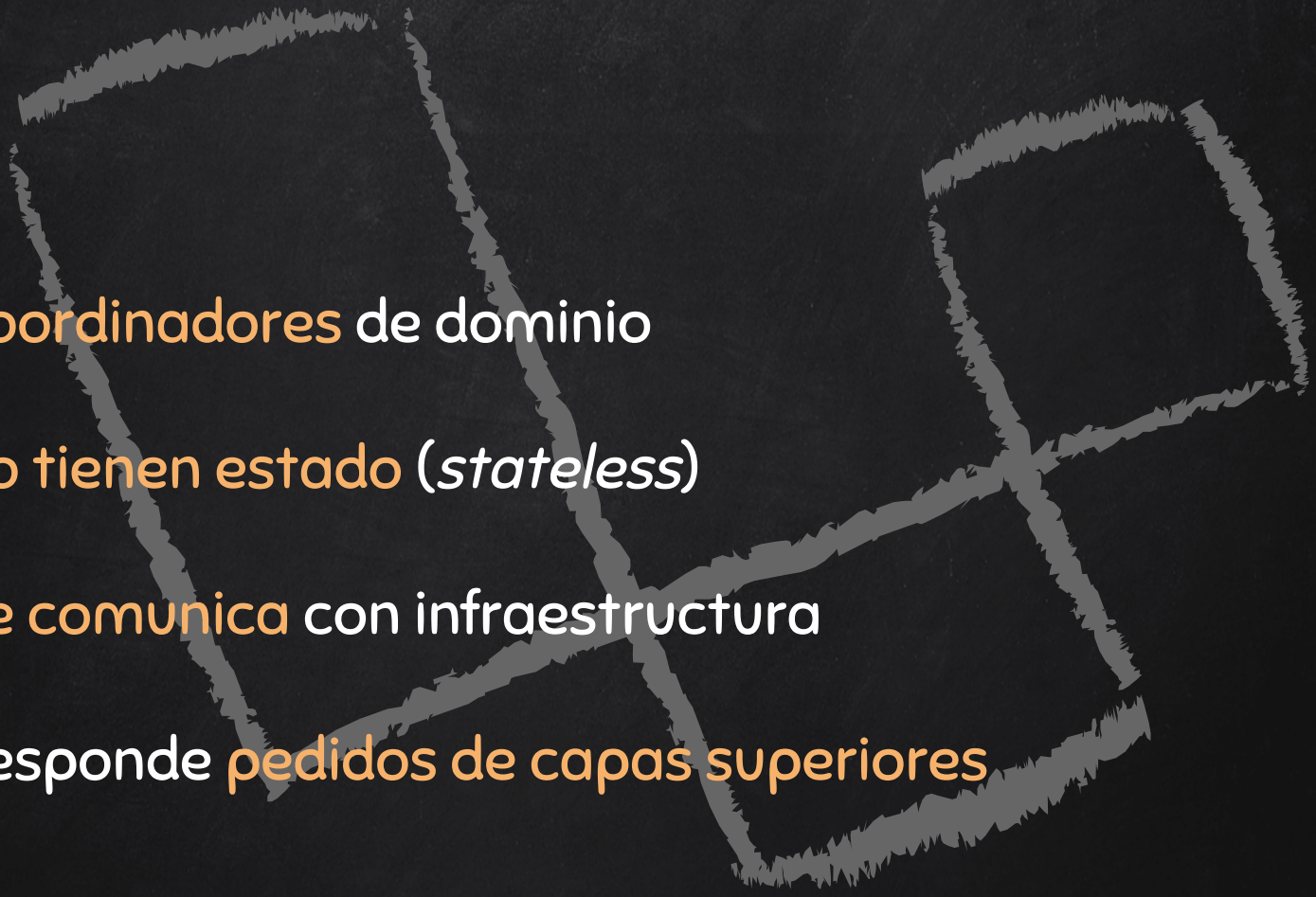
Capas (Layers)



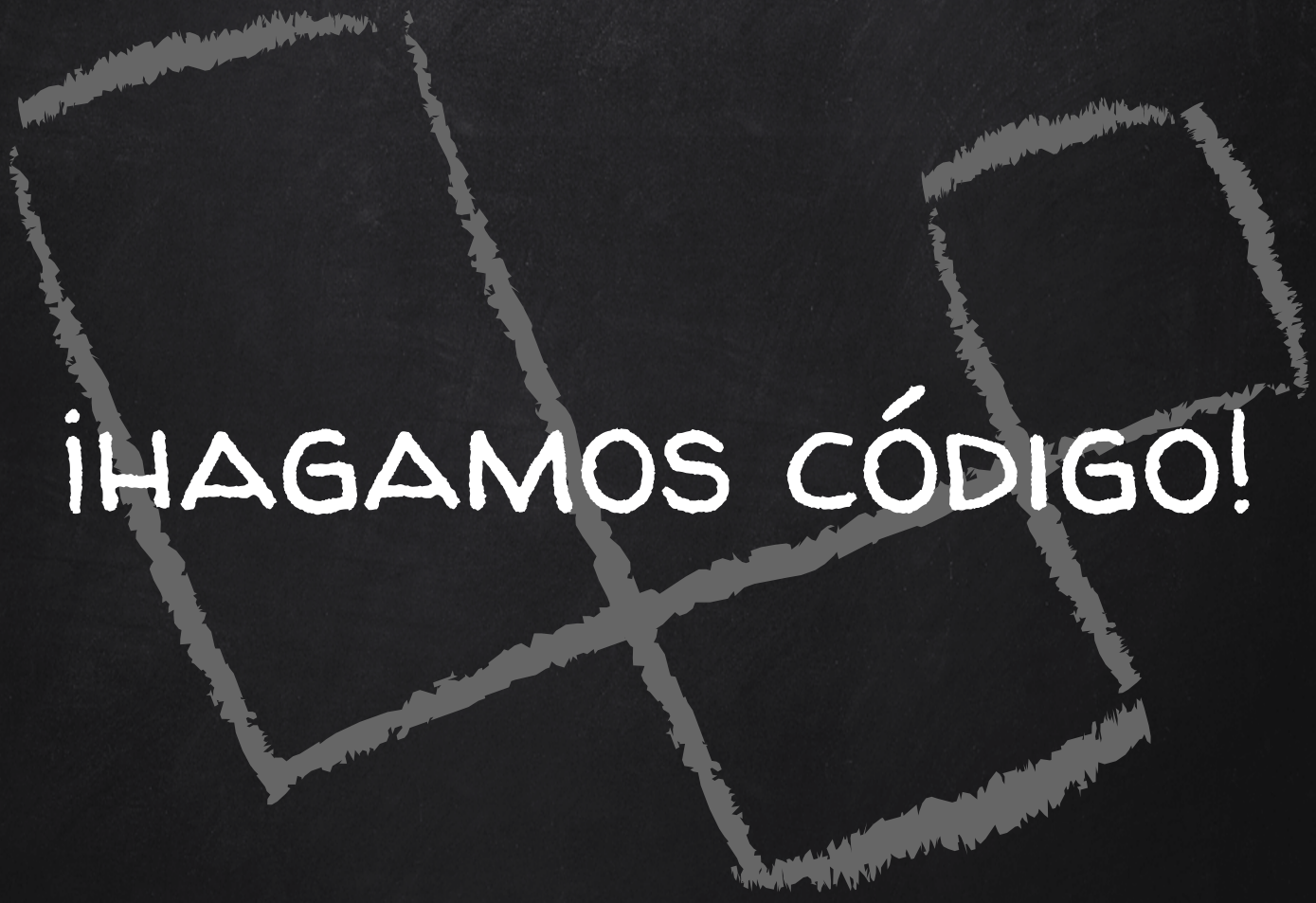

¡HAGAMOS CÓDIGO!

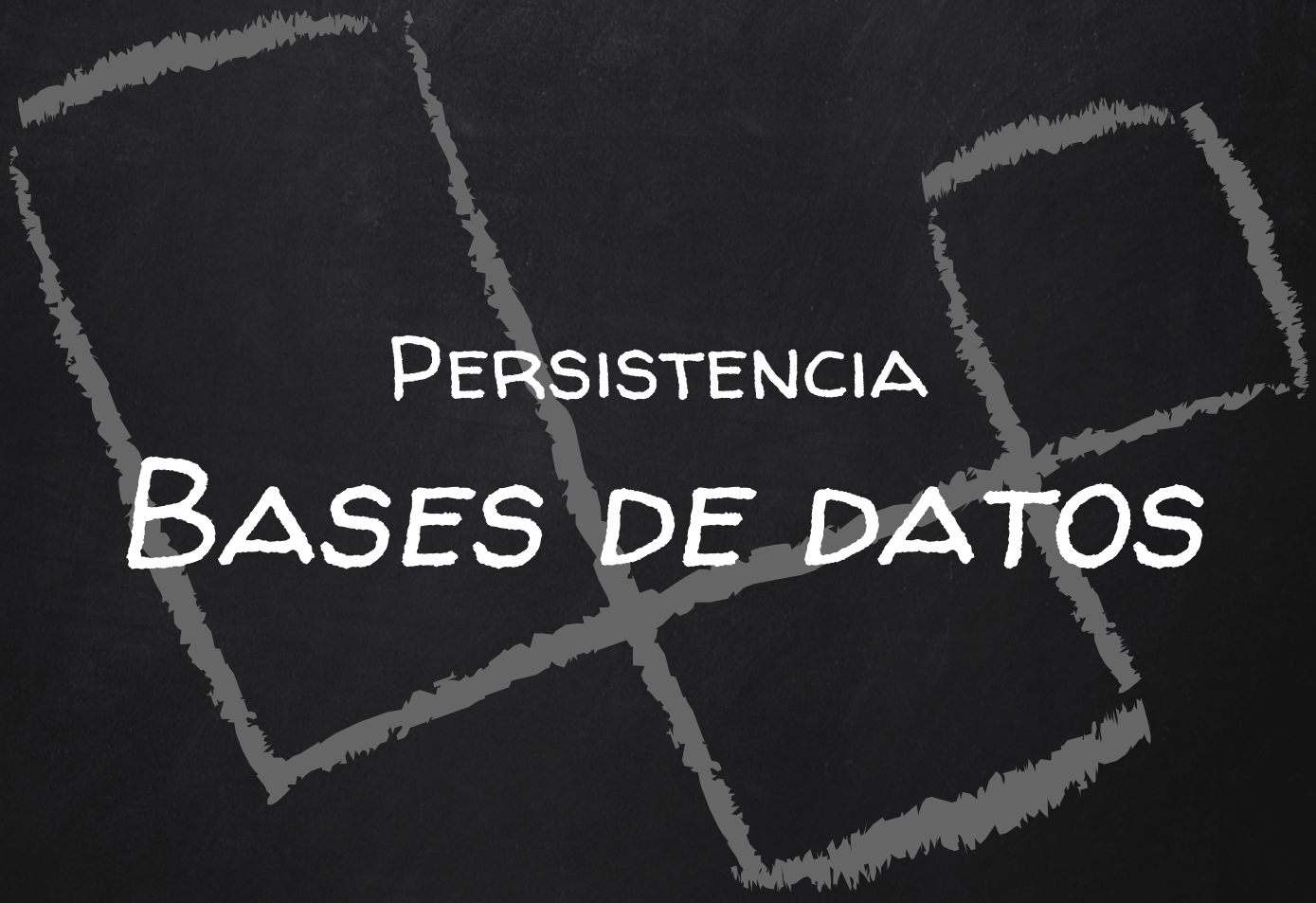

Servicios

Servicios (*Services*)



- ✘ Coordinadores de dominio
- ✘ No tienen estado (*stateless*)
- ✘ Se comunica con infraestructura
- ✘ Responde pedidos de capas superiores

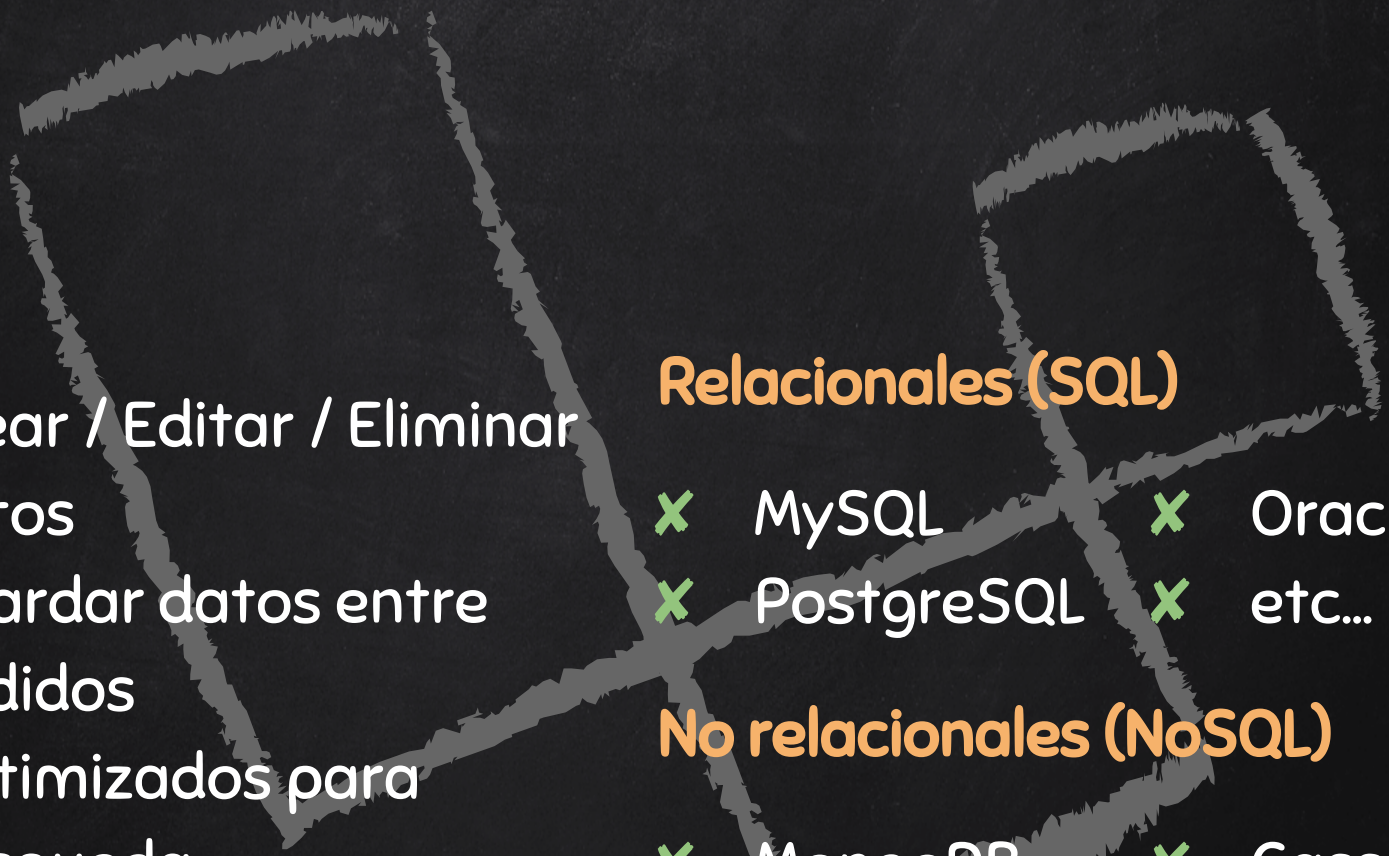

¡HAGAMOS CÓDIGO!



Persistencia

Bases de datos

Bases de datos



- Crear / Editar / Eliminar datos
- Guardar datos entre pedidos
- Optimizados para búsqueda
- Transaccionalidad

Relacionales (SQL)

- MySQL
- PostgreSQL
- Oracle
- etc...

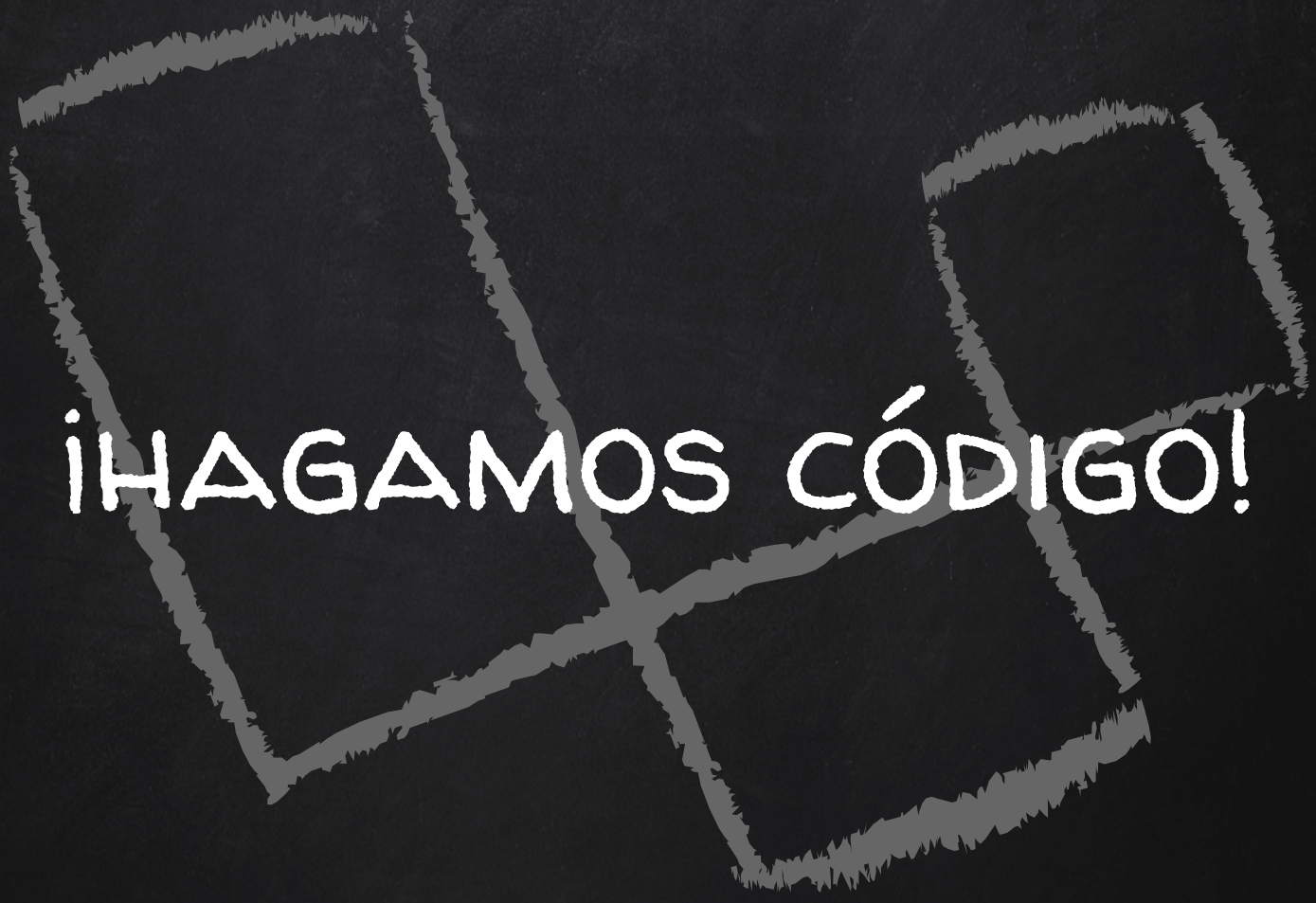
No relacionales (NoSQL)

- MongoDB
- Redis
- Cassandra
- etc...

Bases de datos

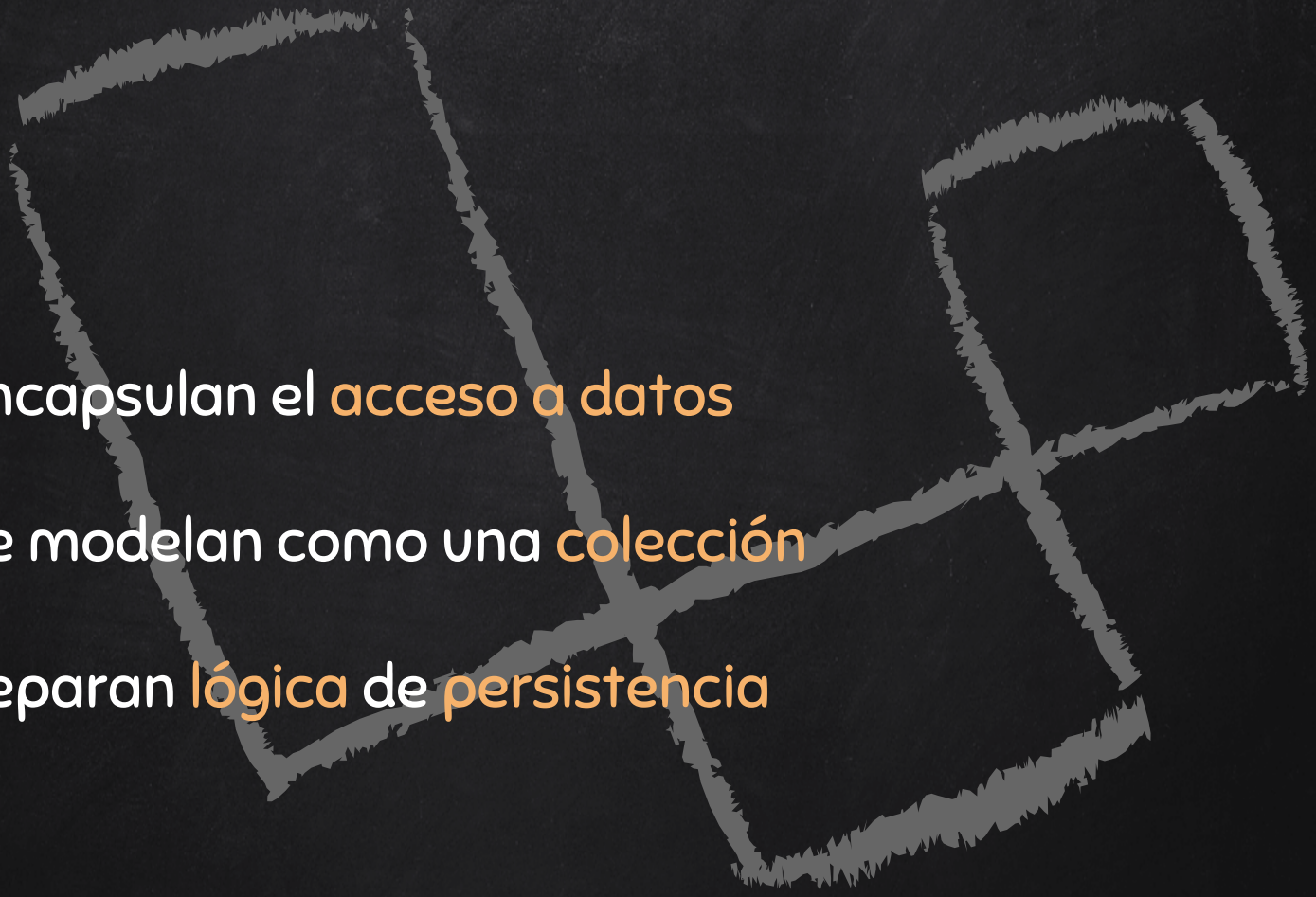
PDO – PHP Data Objects

- ✘ Abstracción de base de datos
- ✘ Modelo de objetos
- ✘ Estándar recomendado por la comunidad

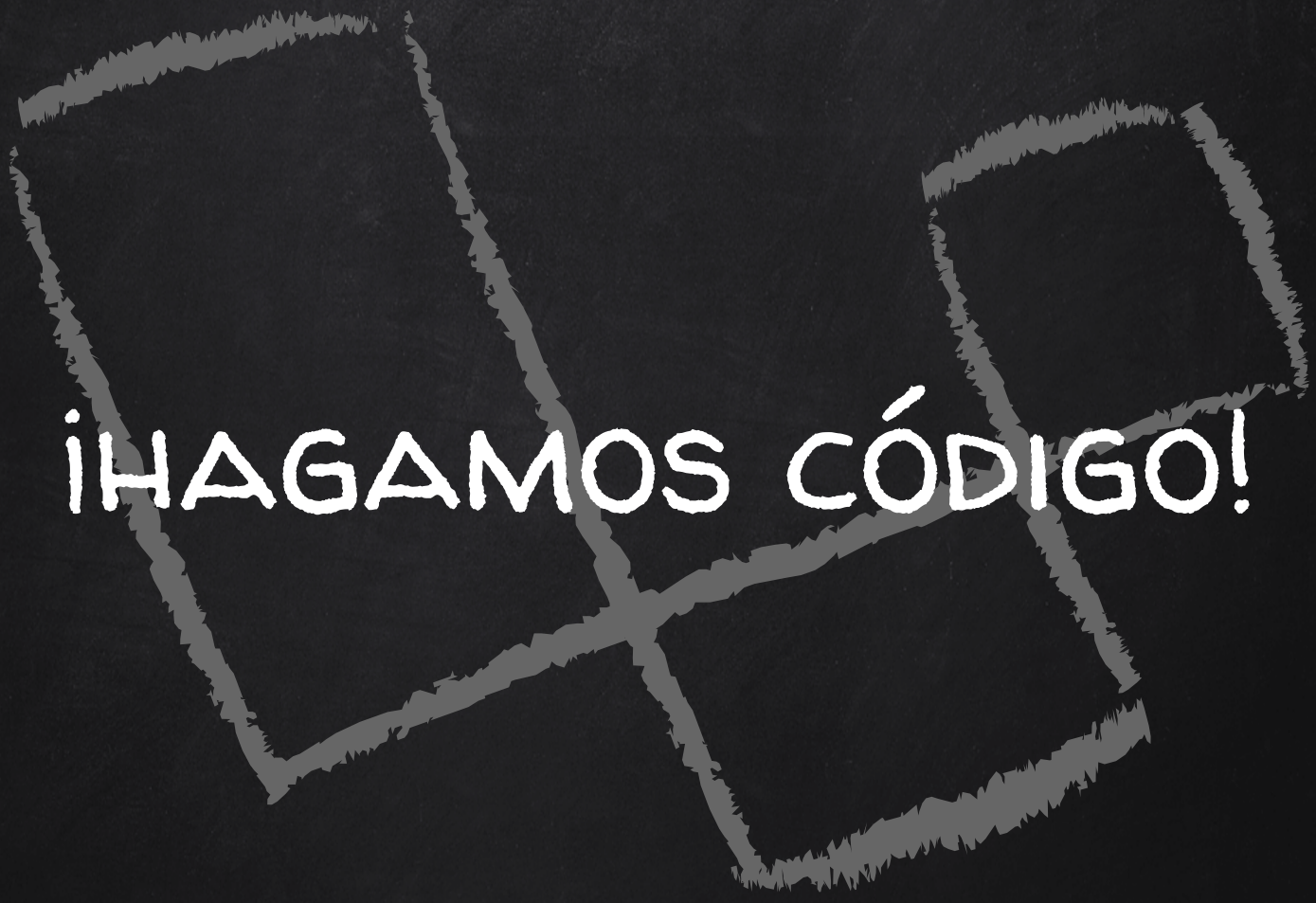

¡HAGAMOS CÓDIGO!


Repositorios

Repositorios (*Repositories*)

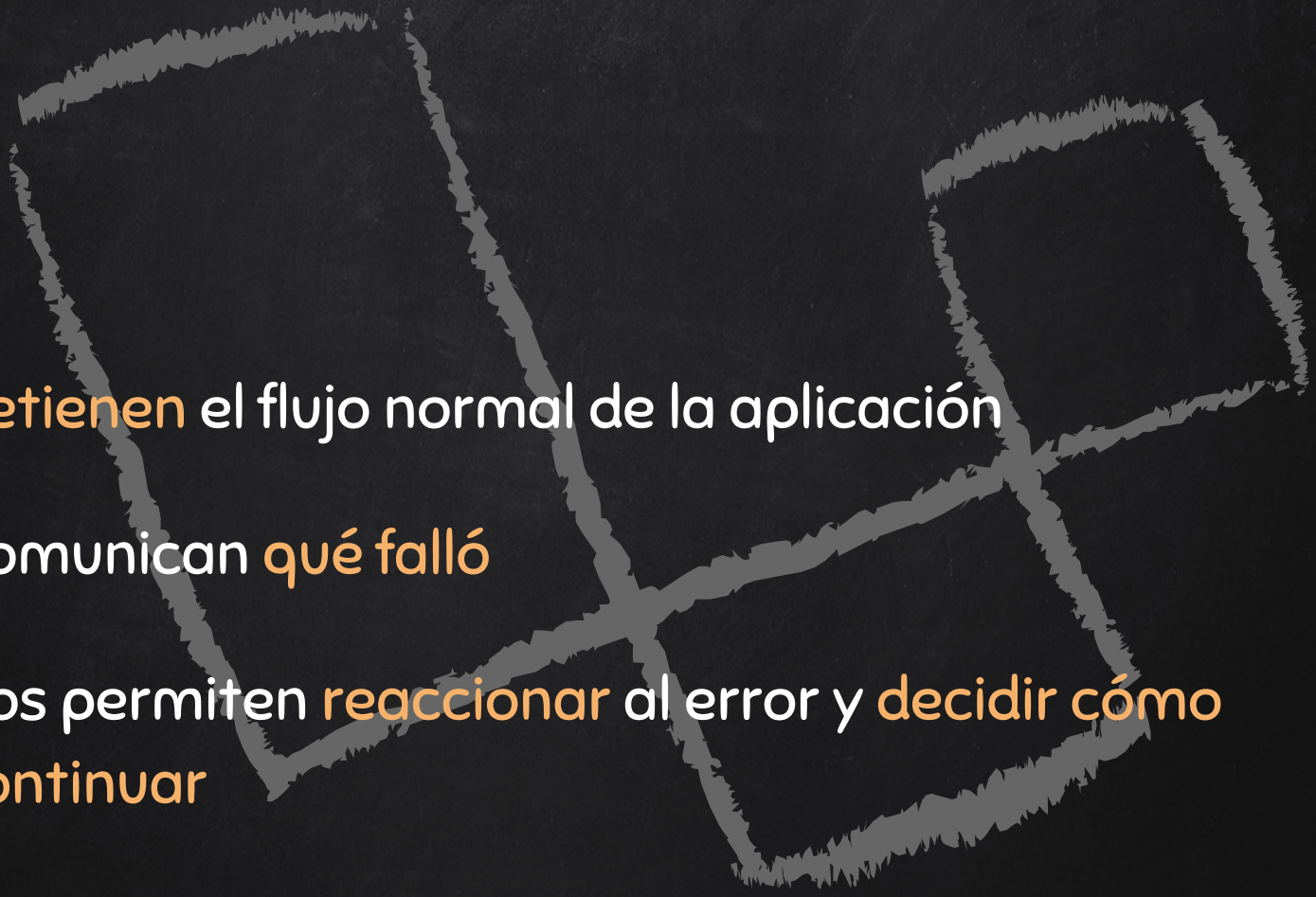


- ✘ Encapsulan el acceso a datos
- ✘ Se modelan como una colección
- ✘ Separan lógica de persistencia

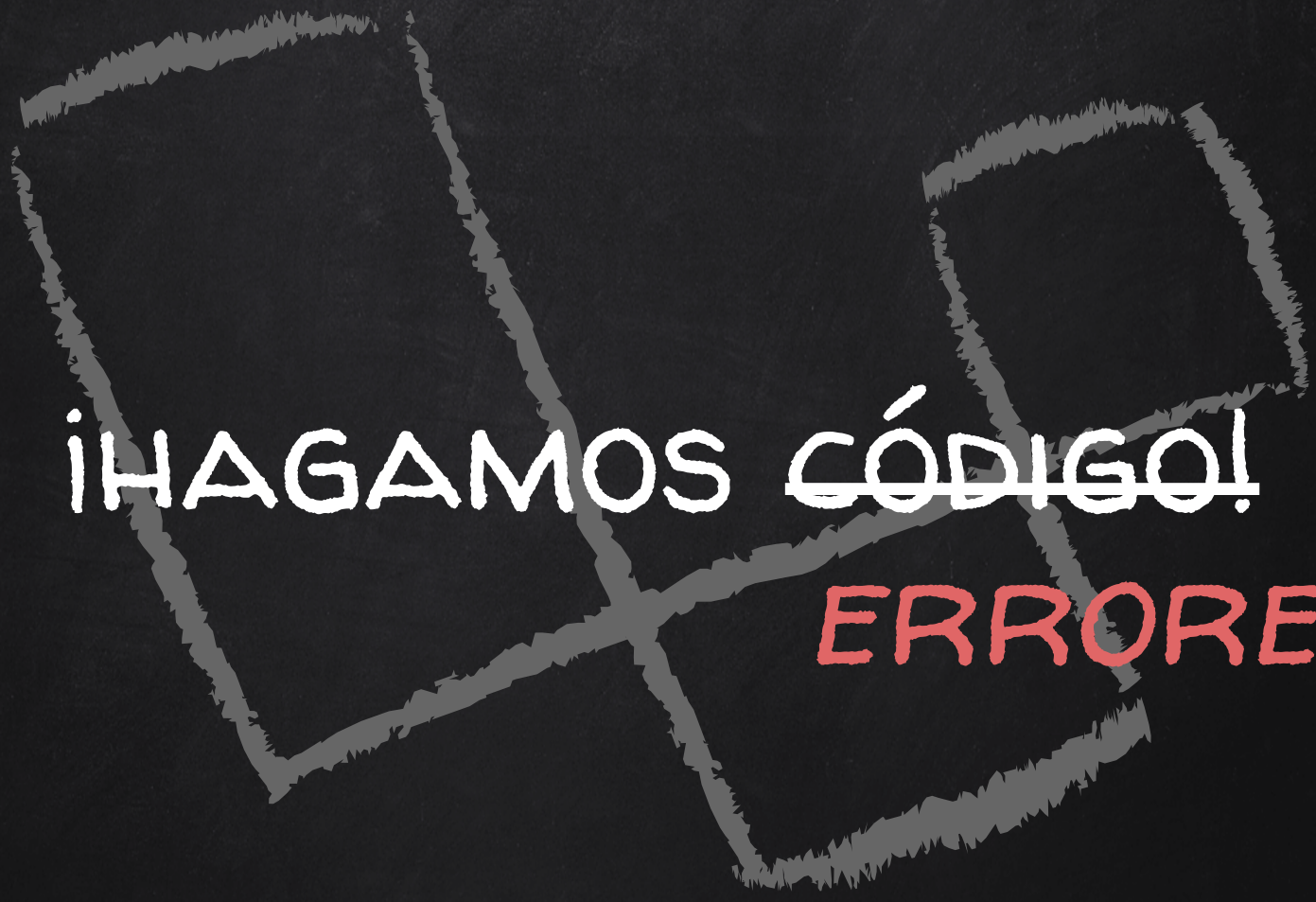

¡HAGAMOS CÓDIGO!


Errores y
Excepciones

Errores y Excepciones (*Exceptions*)

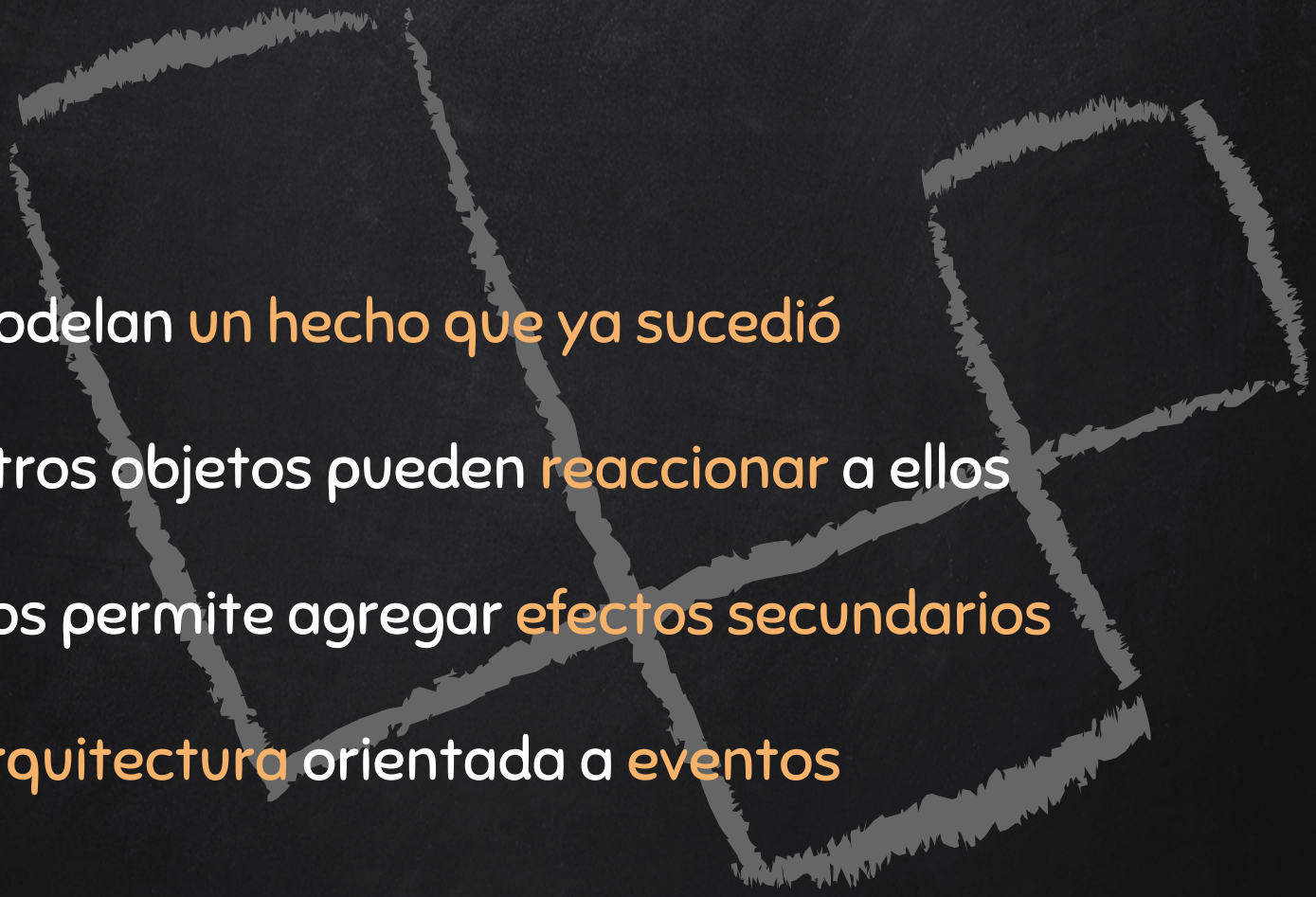


- ✘ Detienen el flujo normal de la aplicación
- ✘ Comunican qué falló
- ✘ Nos permiten reaccionar al error y decidir cómo continuar

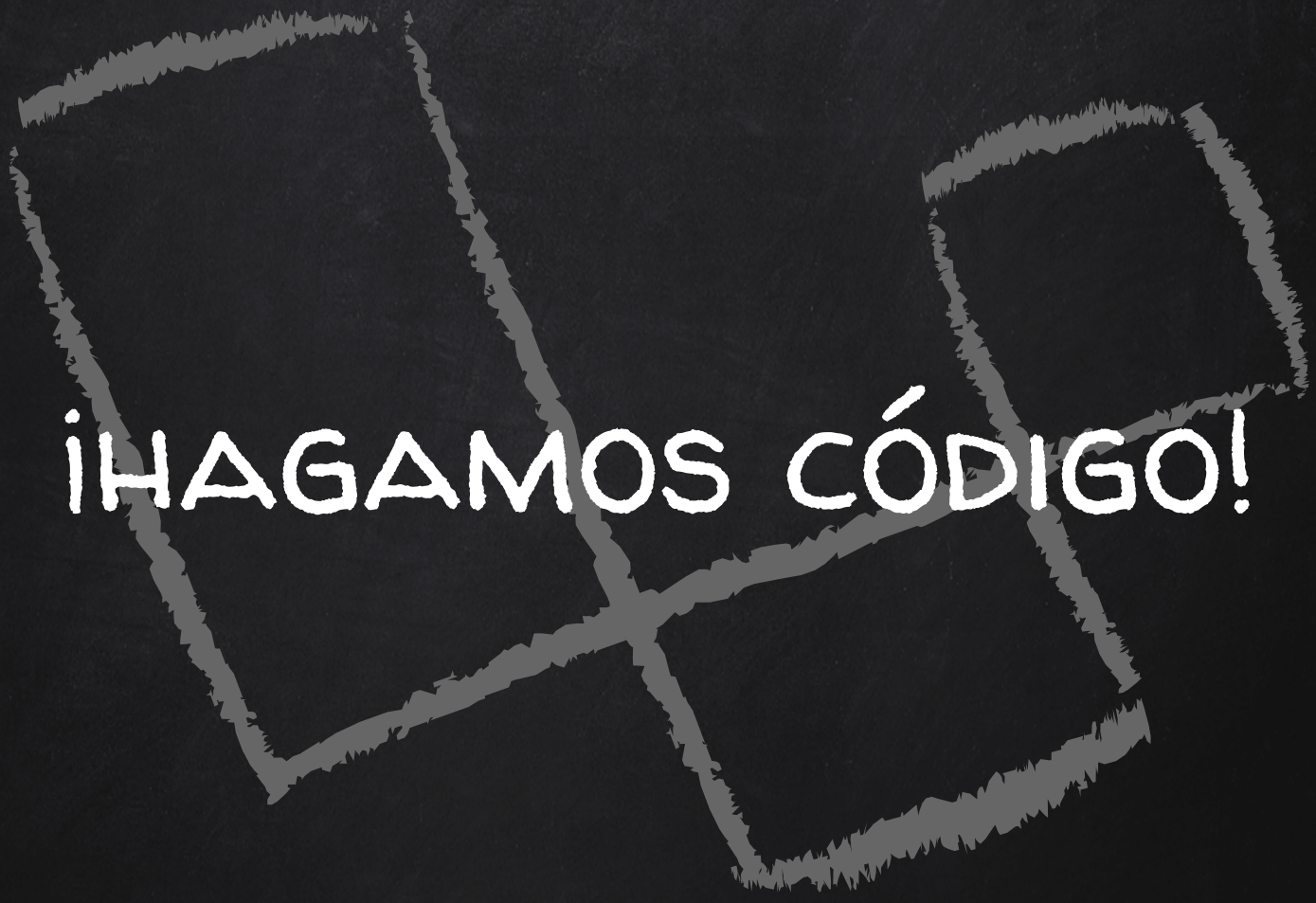

¡HAGAMOS ~~CÓDIGO~~!
ERRORES!


Eventos

Eventos (*Events*)



- Modelan un hecho que ya sucedió
- Otros objetos pueden reaccionar a ellos
- Nos permite agregar efectos secundarios
- Arquitectura orientada a eventos


¡HAGAMOS CÓDIGO!

Resumen

Capas

- ✘ Define roles
- ✘ Separación de arquitectura

Base de datos

- ✘ Almacenar
- ✘ Buscar
- ✘ Persistencia

Excepciones

- ✘ Detiene flujo
- ✘ Recuperación
- ✘ Notificación

Servicios

- ✘ Coordinación
- ✘ Comunicación
- ✘ Sin estado

Repositorios

- ✘ Acceso datos
- ✘ Colección
- ✘ Separa capas

Eventos

- ✘ Hecho pasado
- ✘ Efectos secundarios

Gracias

@guiwoda

/guiwoda

Laravel

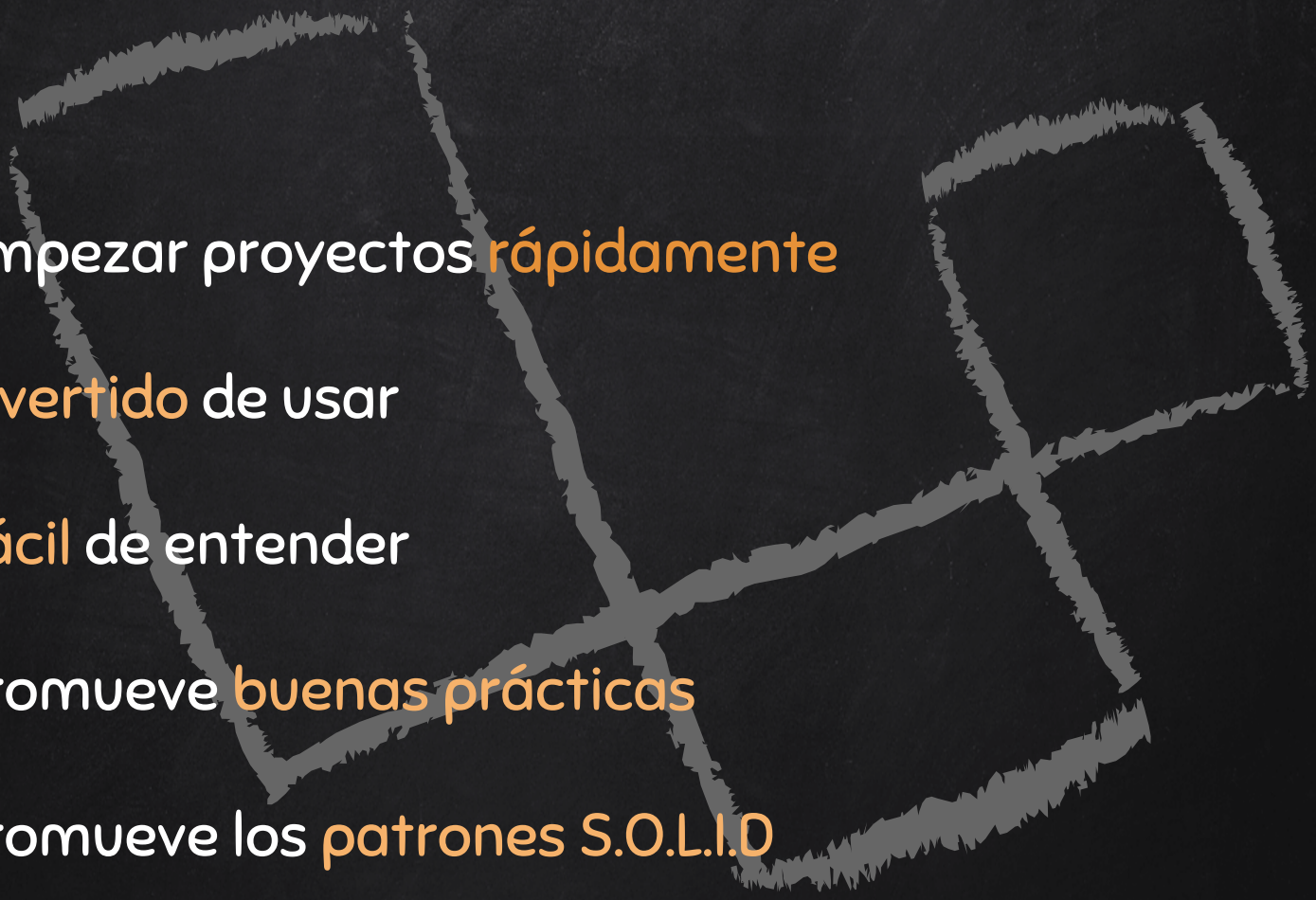
Clase 1
Fundamentos

Joseph Cohen – @joecohens

¿Qué es Laravel?

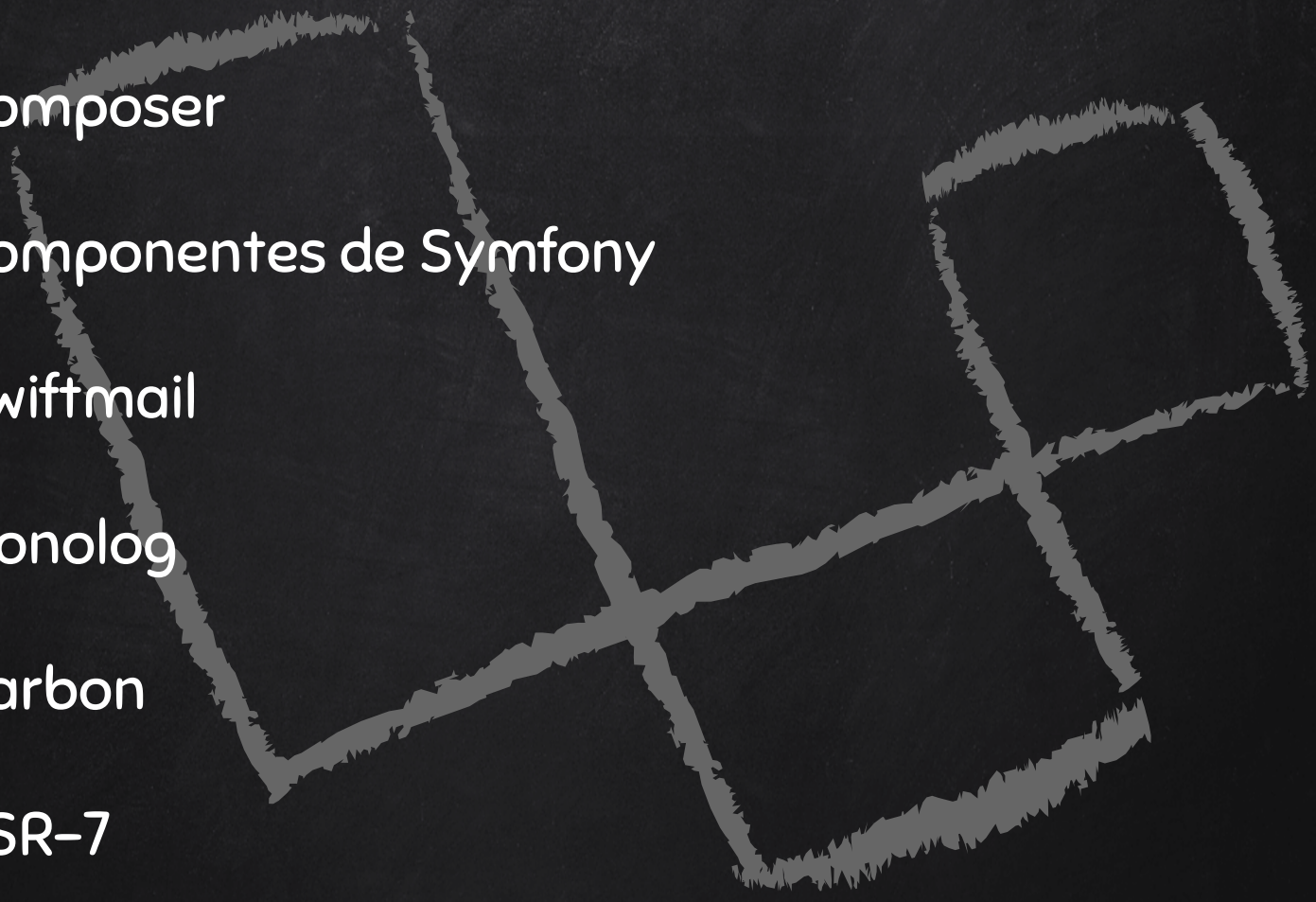
Laravel es un framework moderno de PHP

Objetivos



- ✘ Empezar proyectos rápidamente
- ✘ Divertido de usar
- ✘ Fácil de entender
- ✘ Promueve buenas prácticas
- ✘ Promueve los patrones S.O.L.I.D

La base

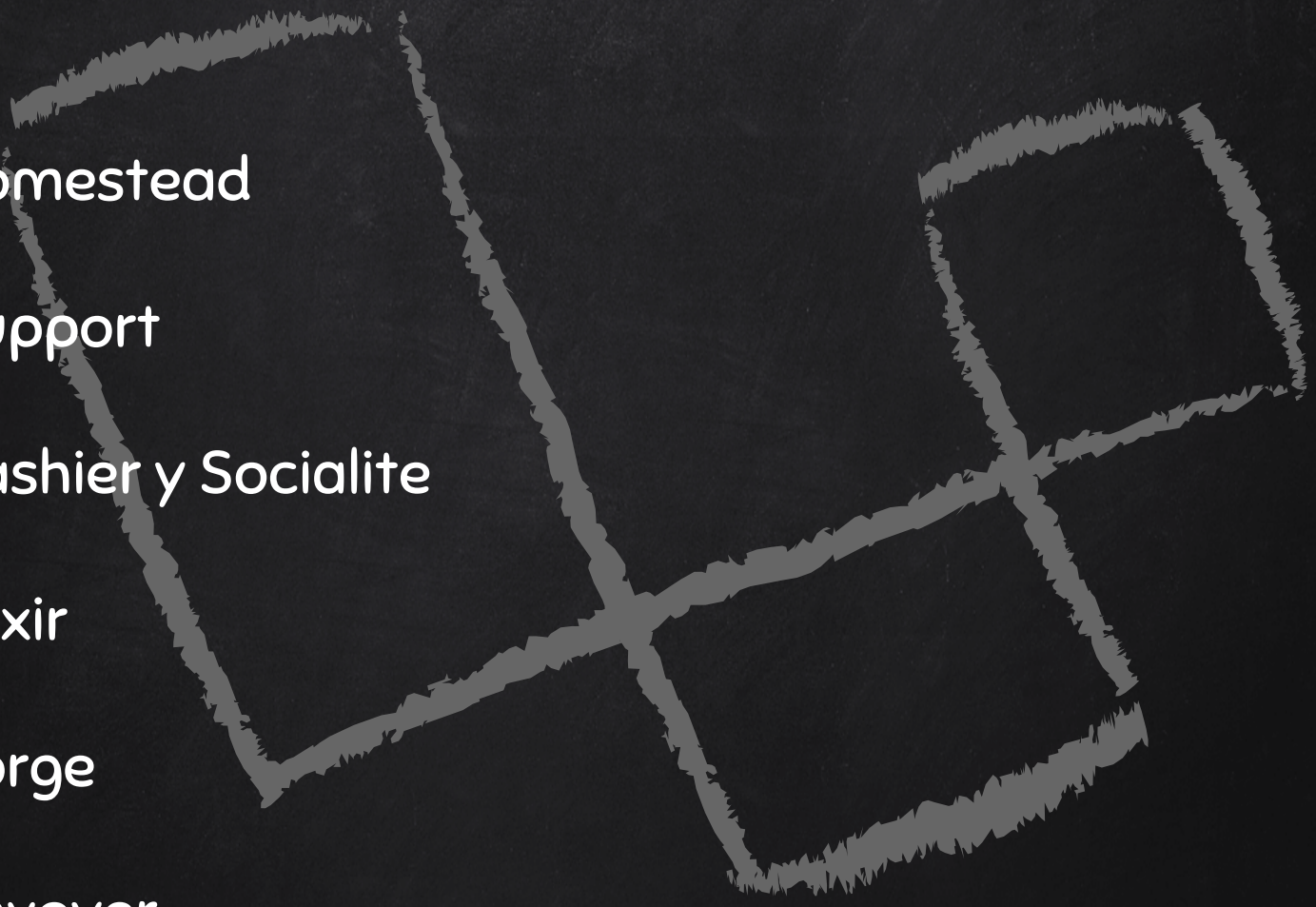


- ✘ Composer
- ✘ Componentes de Symfony
- ✘ Swiftmail
- ✘ Monolog
- ✘ Carbon
- ✘ PSR-7
- ✘ Y más ...

Principales Características

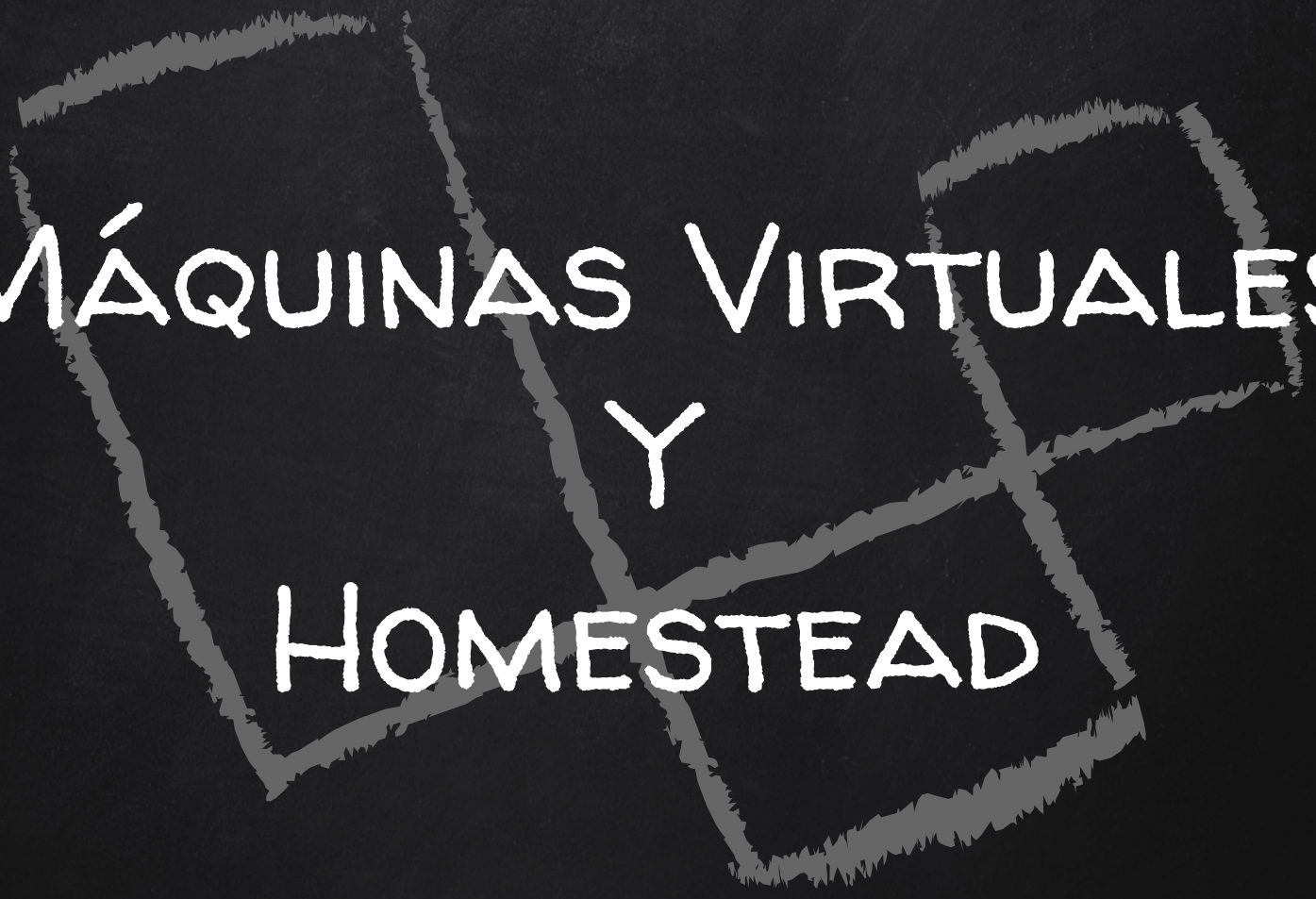
- ✘ Artisan CLI
- ✘ Controladores
- ✘ Eloquent ORM
- ✘ Migraciones
- ✘ Blade templates
- ✘ Poderoso Router
- ✘ Cache
- ✘ Eventos
- ✘ Autenticación
- ✘ Queues
- ✘ ACL
- ✘ Poderoso contenedor

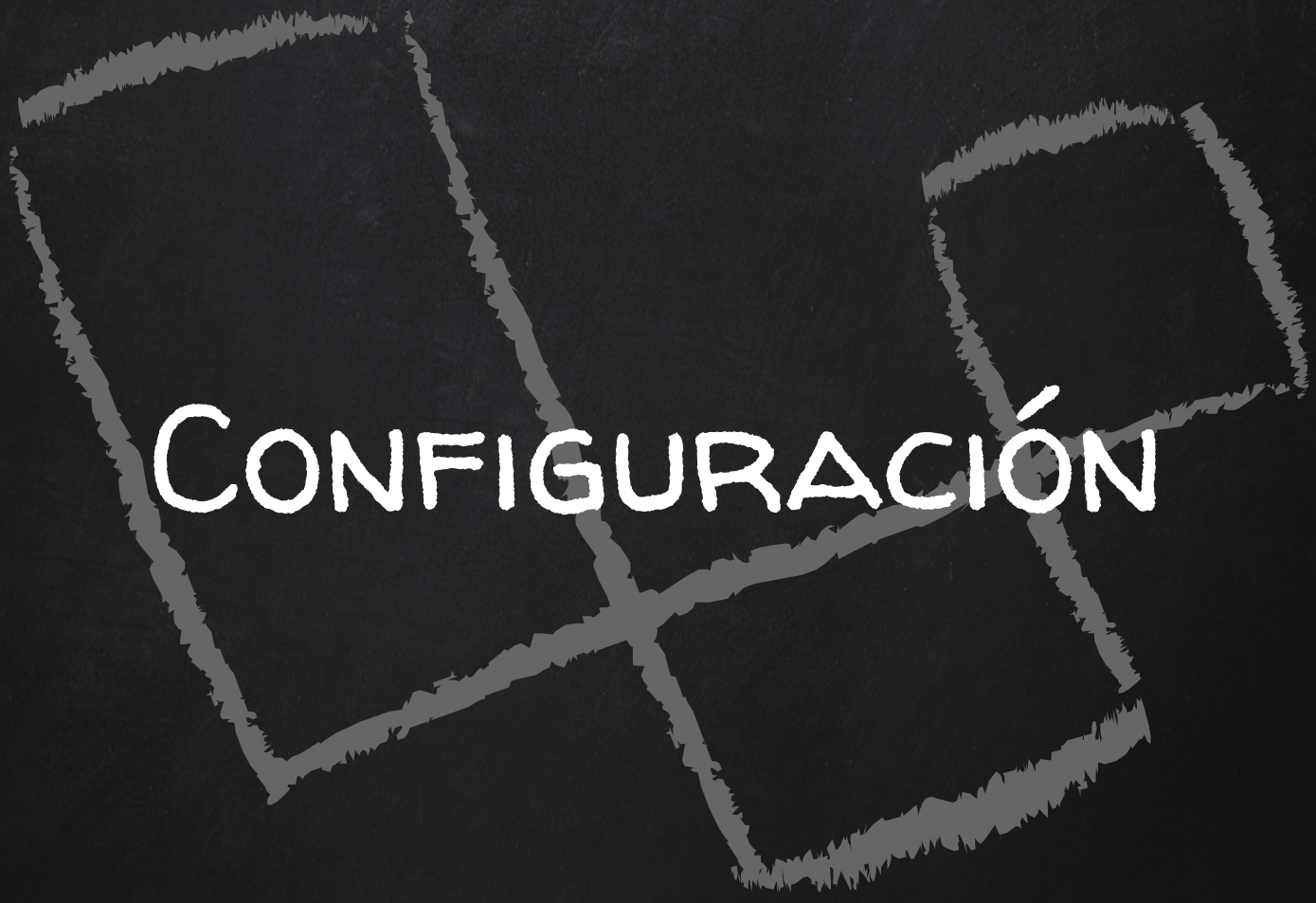
Ecosistema



- ✘ Homestead
- ✘ Support
- ✘ Cashier y Socialite
- ✘ Elixir
- ✘ Forge
- ✘ Envoyer


INSTALACIÓN


Máquinas Virtuales
y
Homestead


Configuración


Rutas

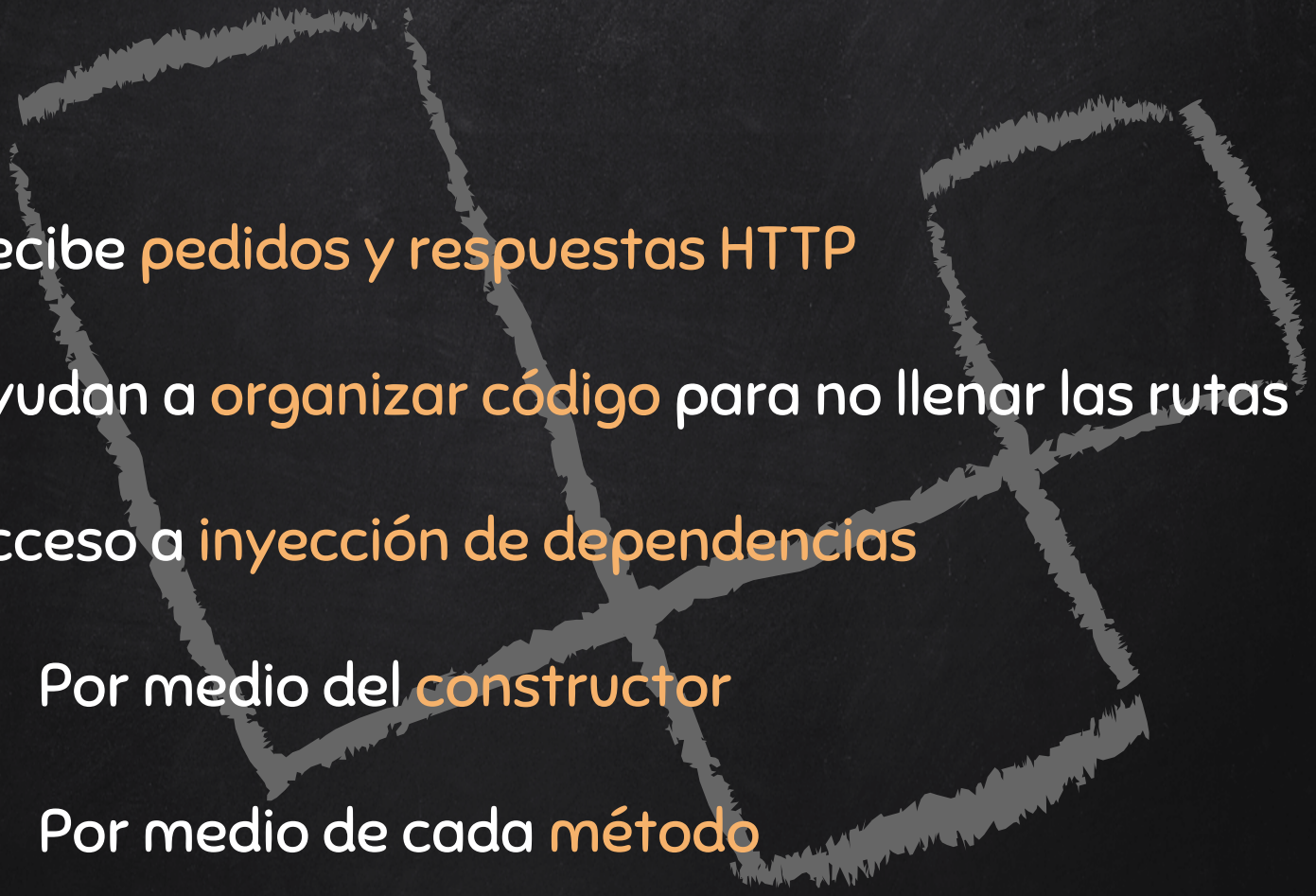
Rutas

- ✘ Métodos HTTP (GET, POST, PUT, PATCH, DELETE)
- ✘ Configuración de parámetros
  - Requeridos
  - Opcionales
  - Regex
- ✘ Agrupadas
- ✘ Por nombre

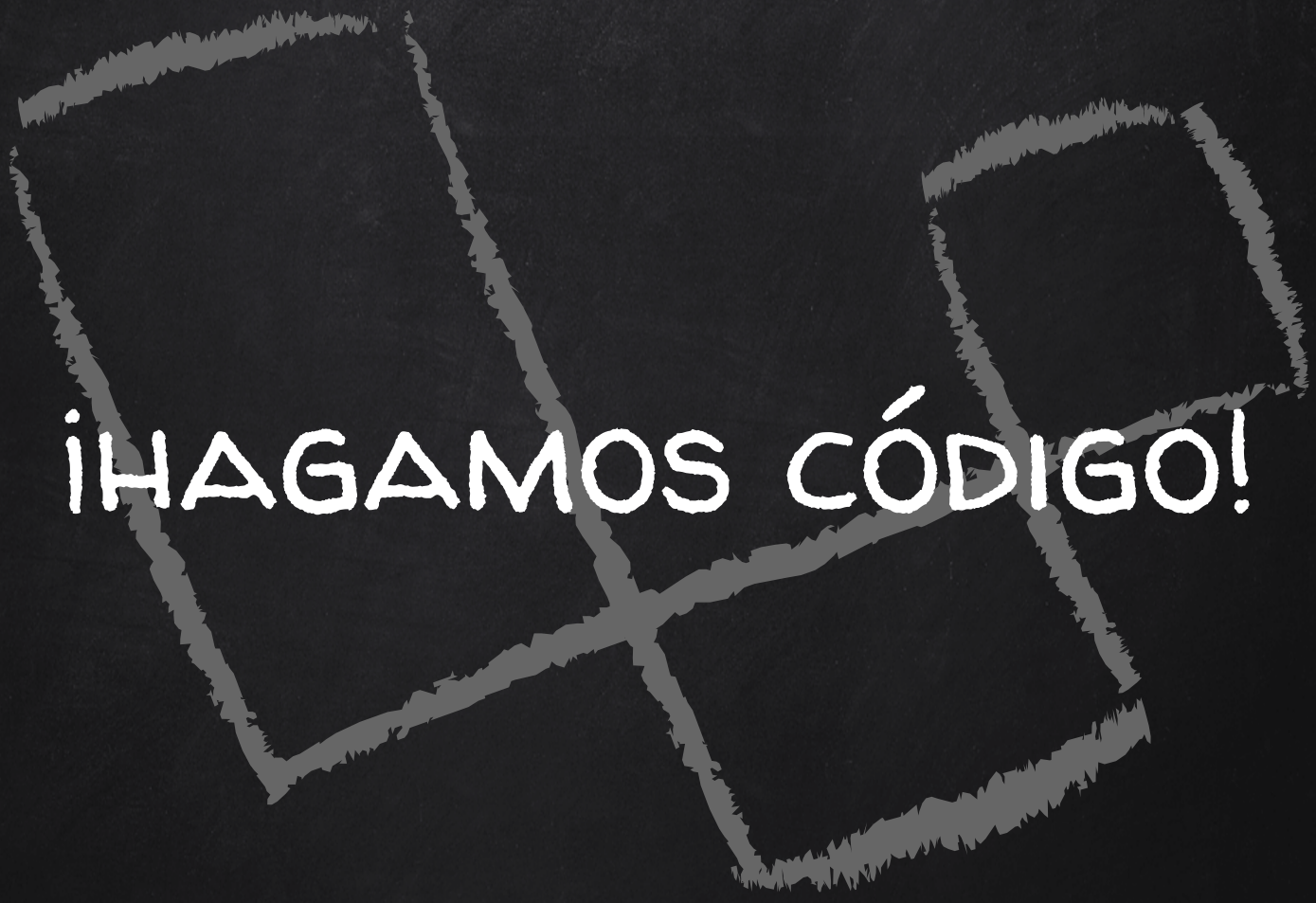

¡HAGAMOS RUTAS!

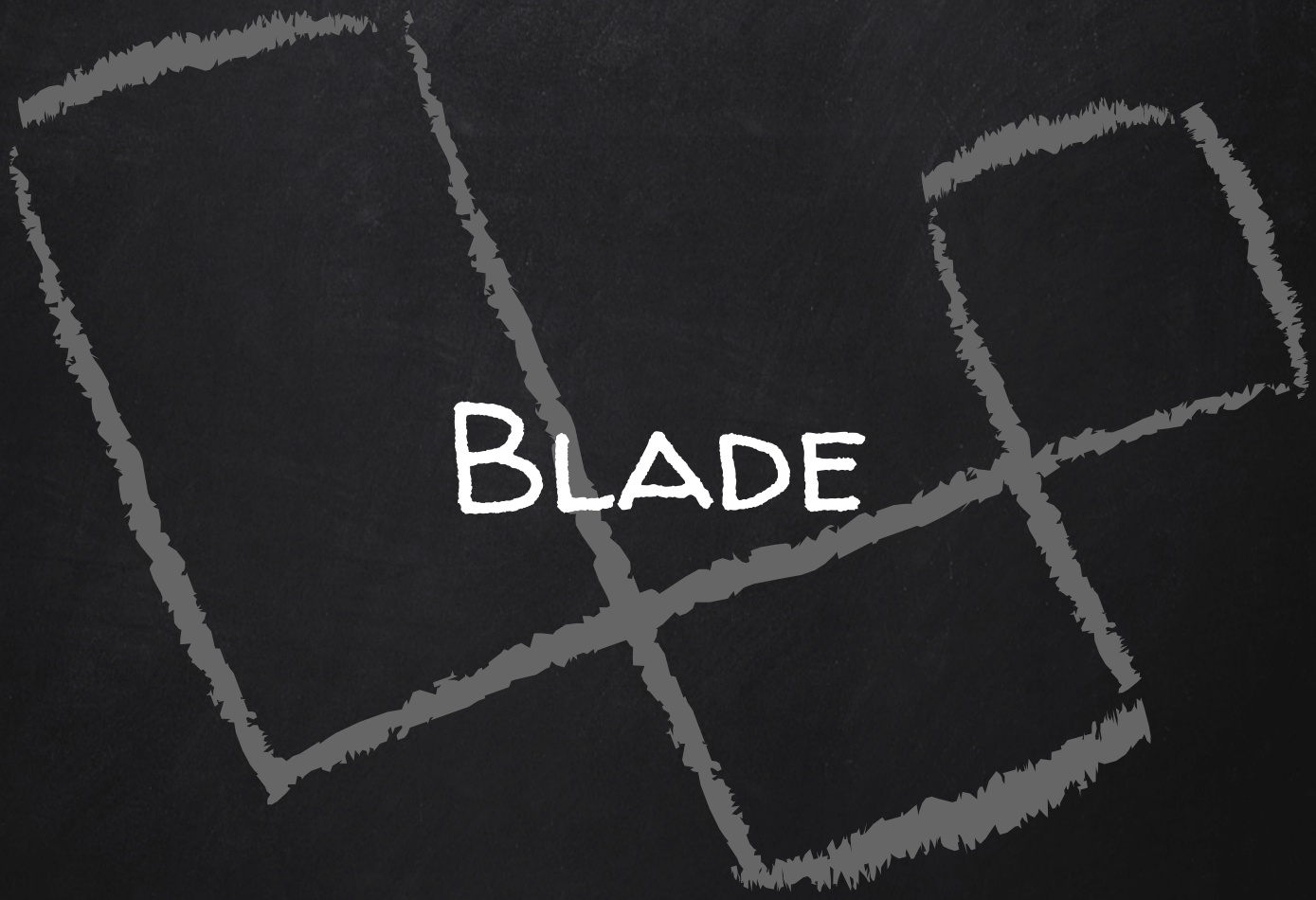

Controladores

Controladores

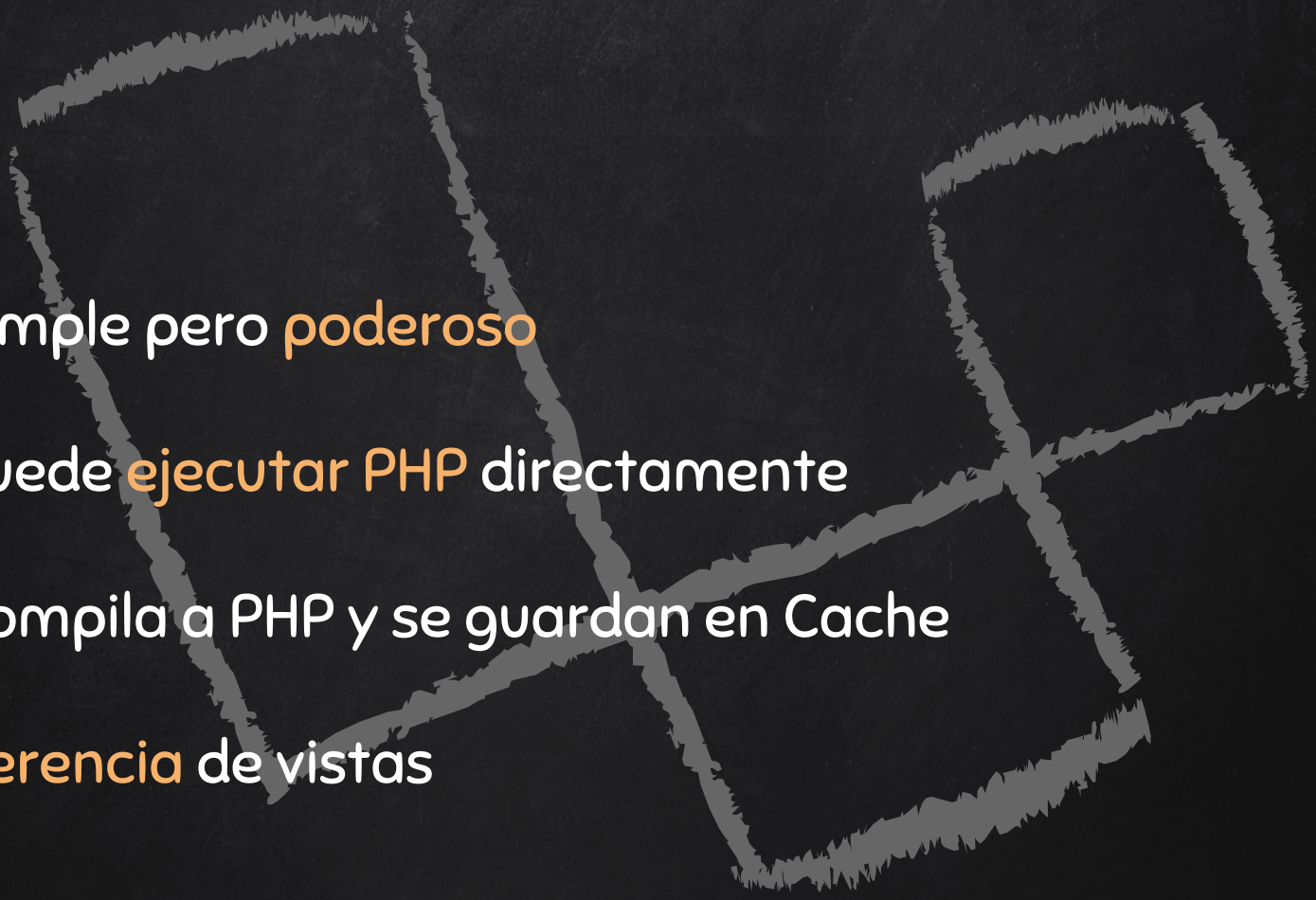


- ✘ Recibe pedidos y respuestas HTTP
- ✘ Ayudan a organizar código para no llenar las rutas
- ✘ Acceso a inyección de dependencias
  - Por medio del constructor
  - Por medio de cada método


¡HAGAMOS CÓDIGO!

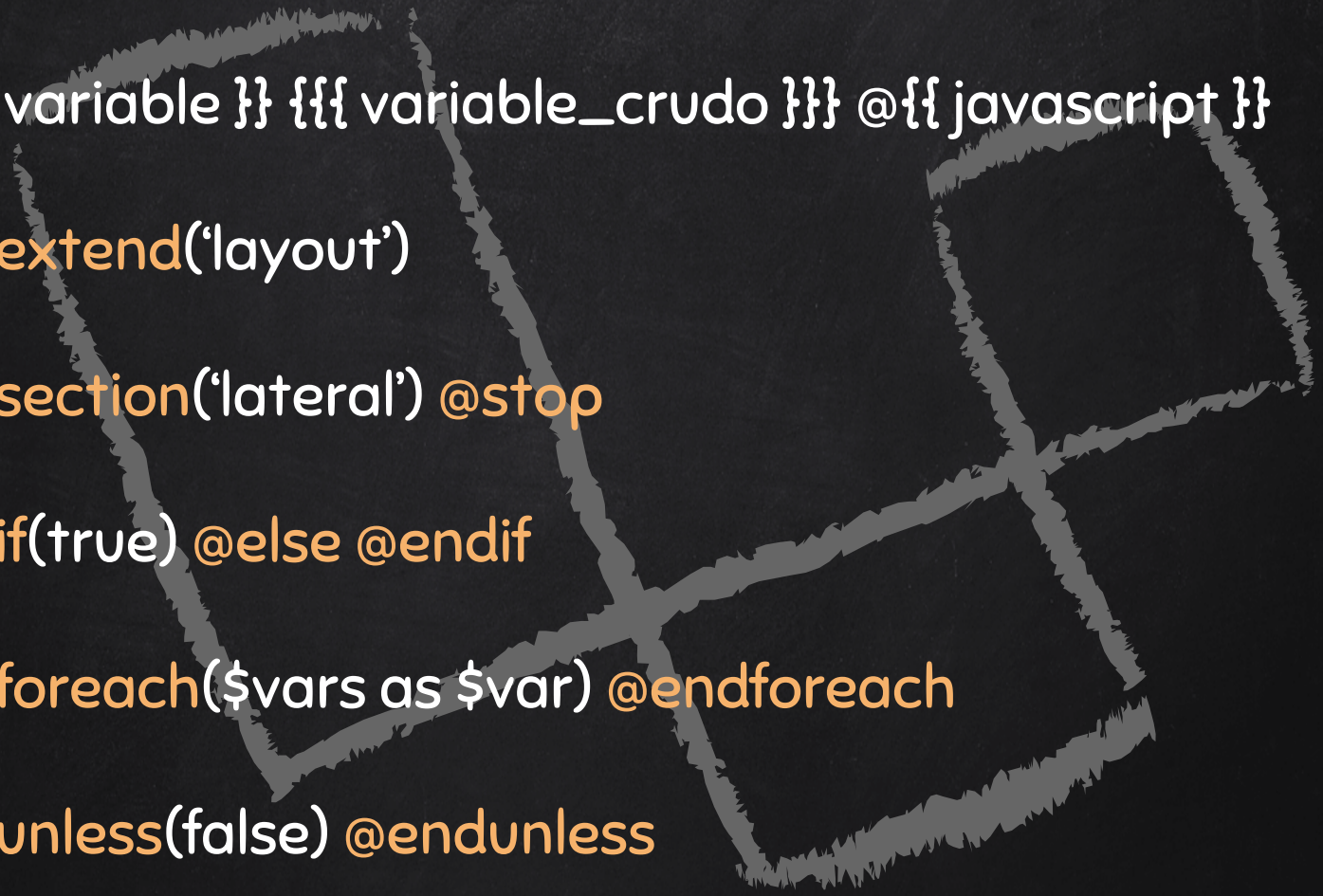

Blade

Blade

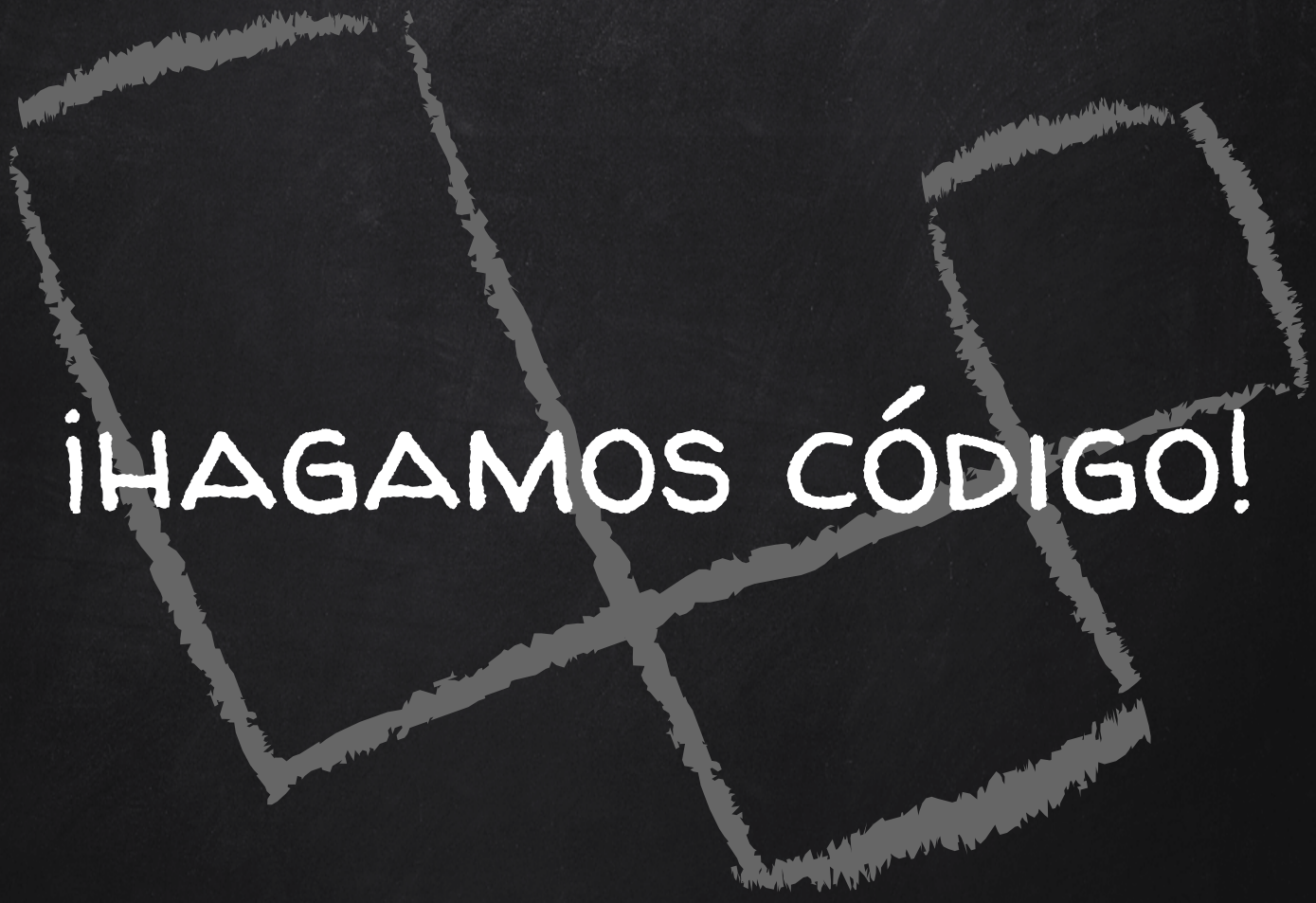


- ✗ Simple pero poderoso
- ✗ Puede ejecutar PHP directamente
- ✗ Compila a PHP y se guardan en Cache
- ✗ Herencia de vistas

Sintaxis



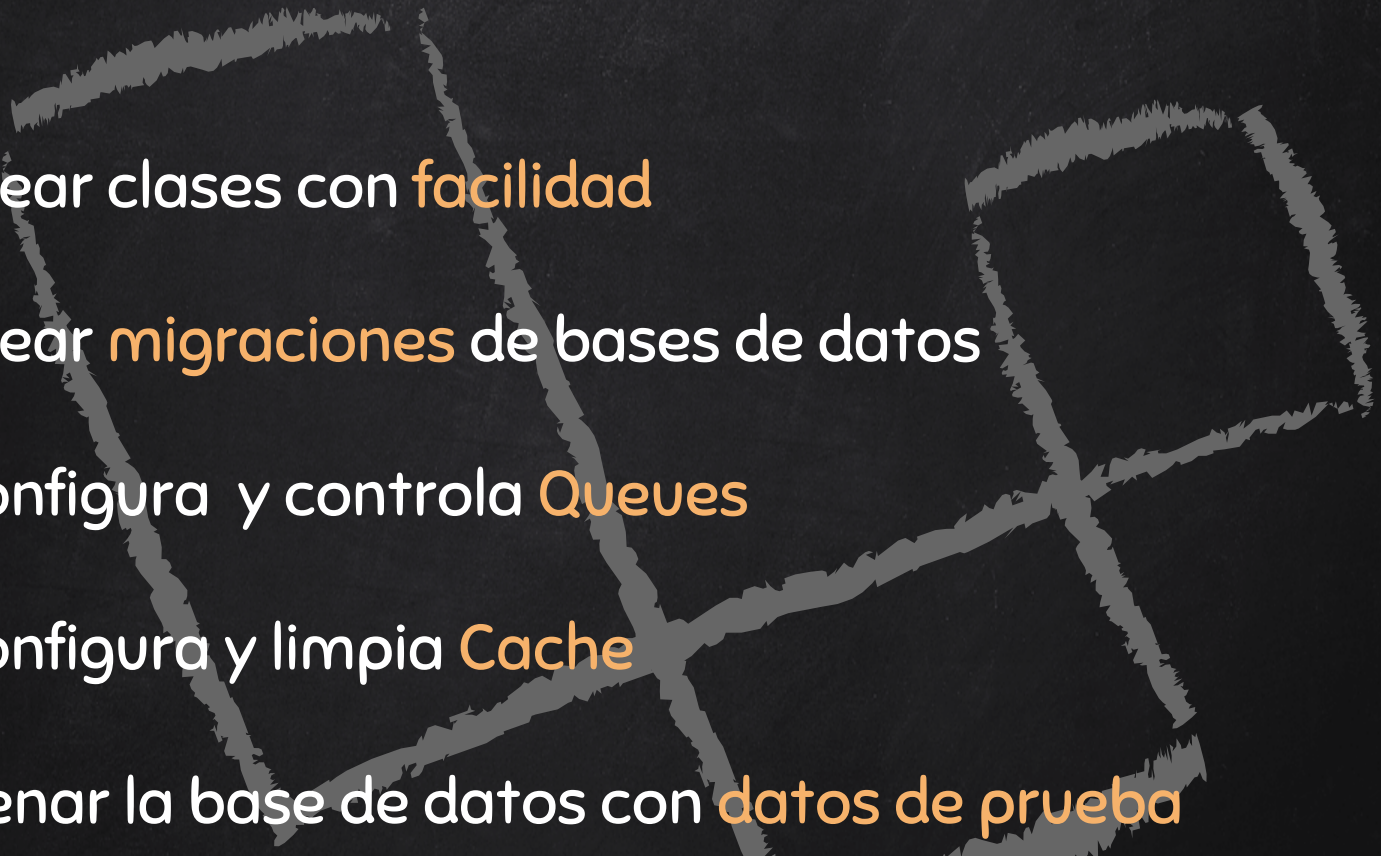
- ✘ {{ variable }} {{{ variable_crudo }}} @{{ javascript }}
- ✘ @extend('layout')
- ✘ @section('lateral') @stop
- ✘ @if(true) @else @endif
- ✘ @foreach($vars as $var) @endforeach
- ✘ @unless(false) @endunless
- ✘ @include('parcial')


¡HAGAMOS CÓDIGO!


Elixir


Artisan

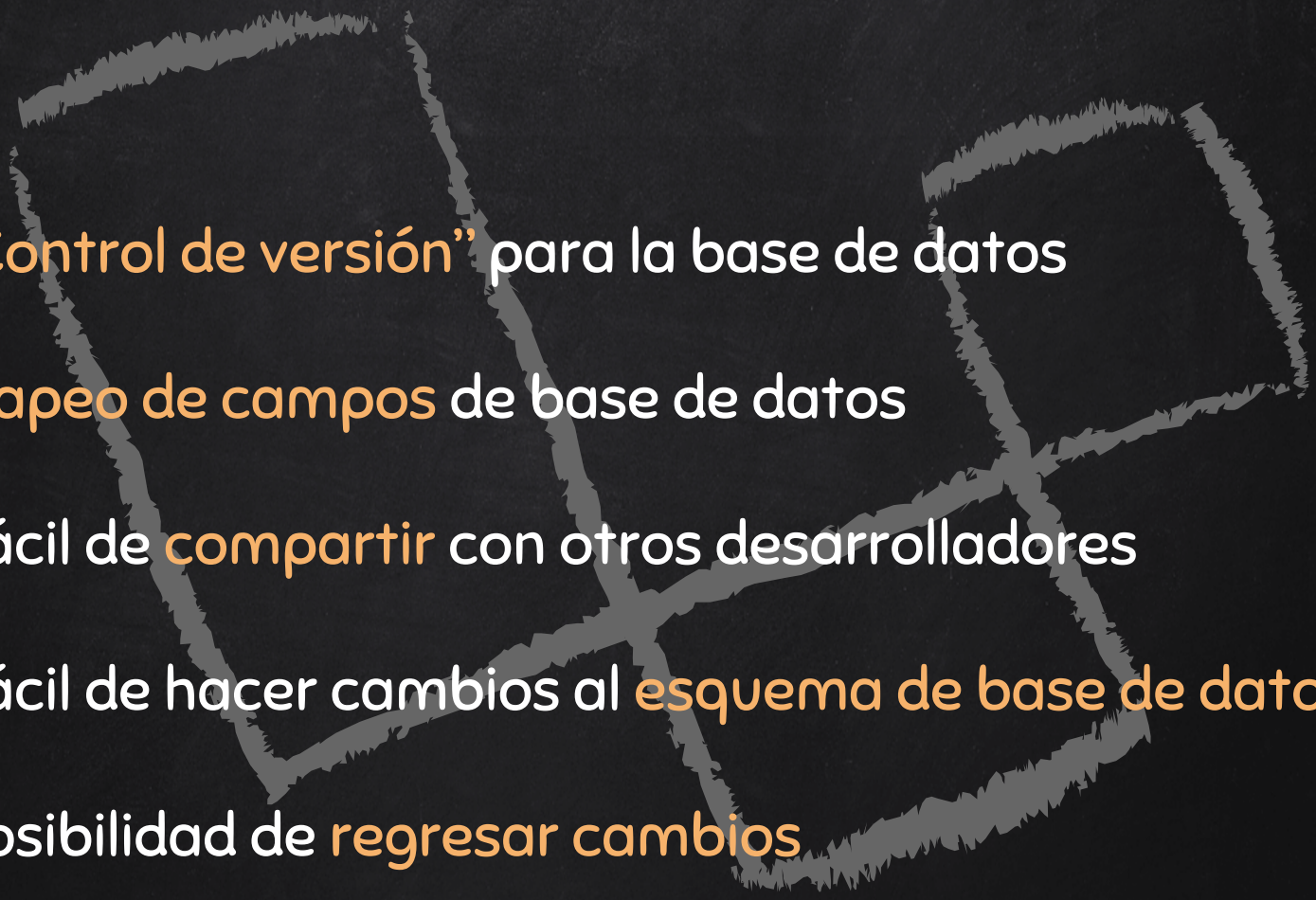
Línea de comando



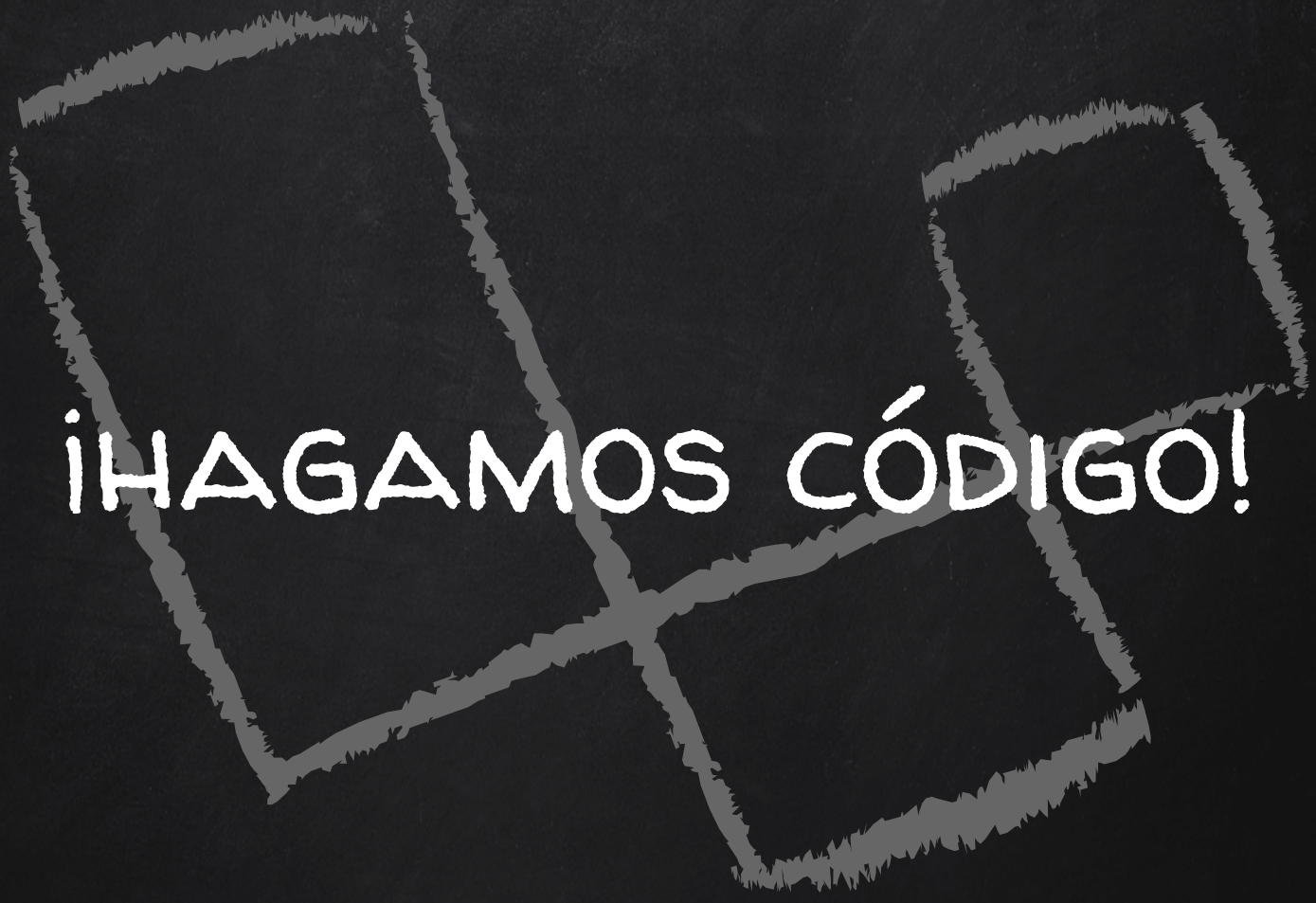
- ✘ Crear clases con facilidad
- ✘ Crear migraciones de bases de datos
- ✘ Configura y controla Queues
- ✘ Configura y limpia Cache
- ✘ Llenar la base de datos con datos de prueba
- ✘ Acceso a la aplicación por medio de la consola

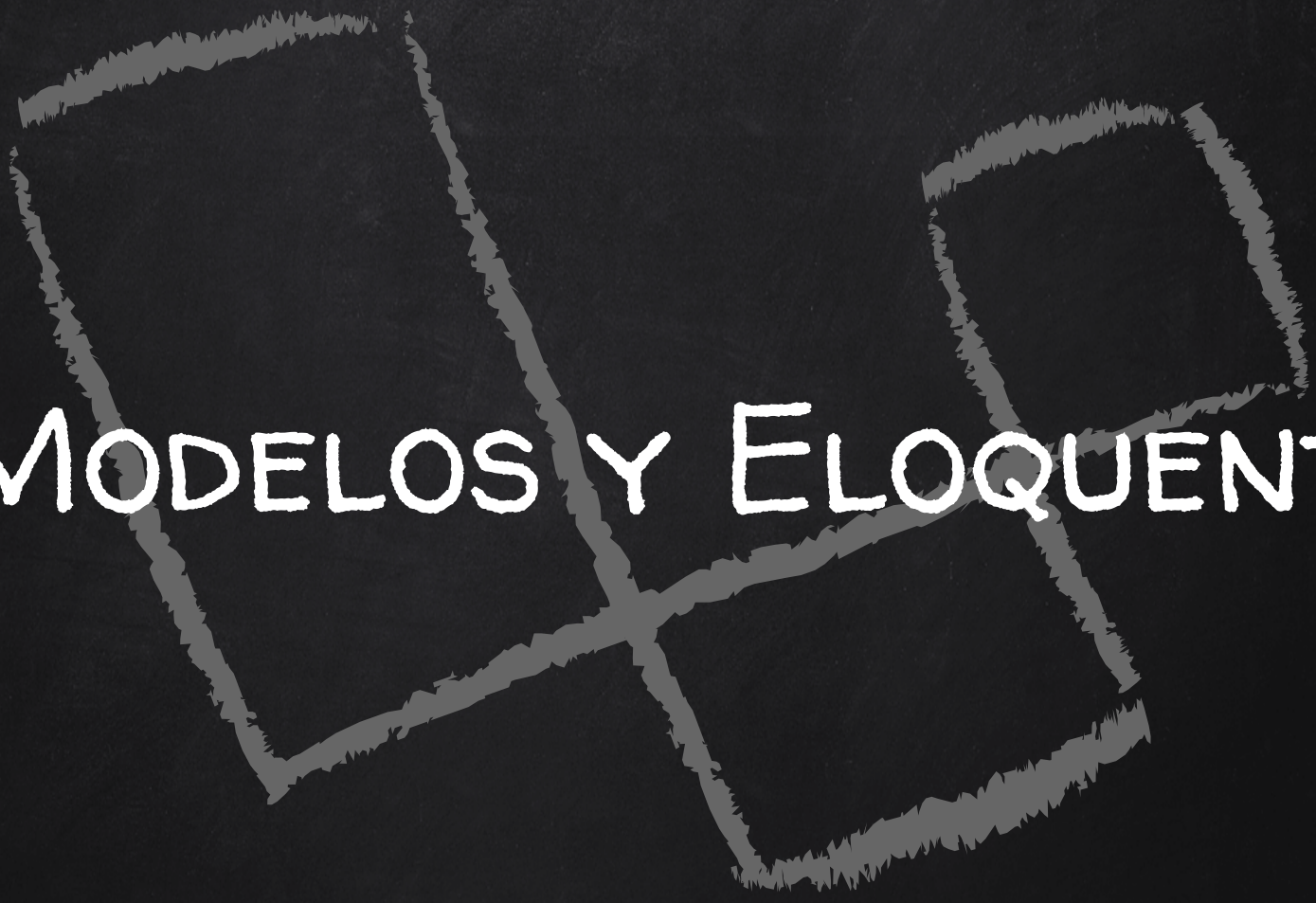

Migraciones

Migraciones

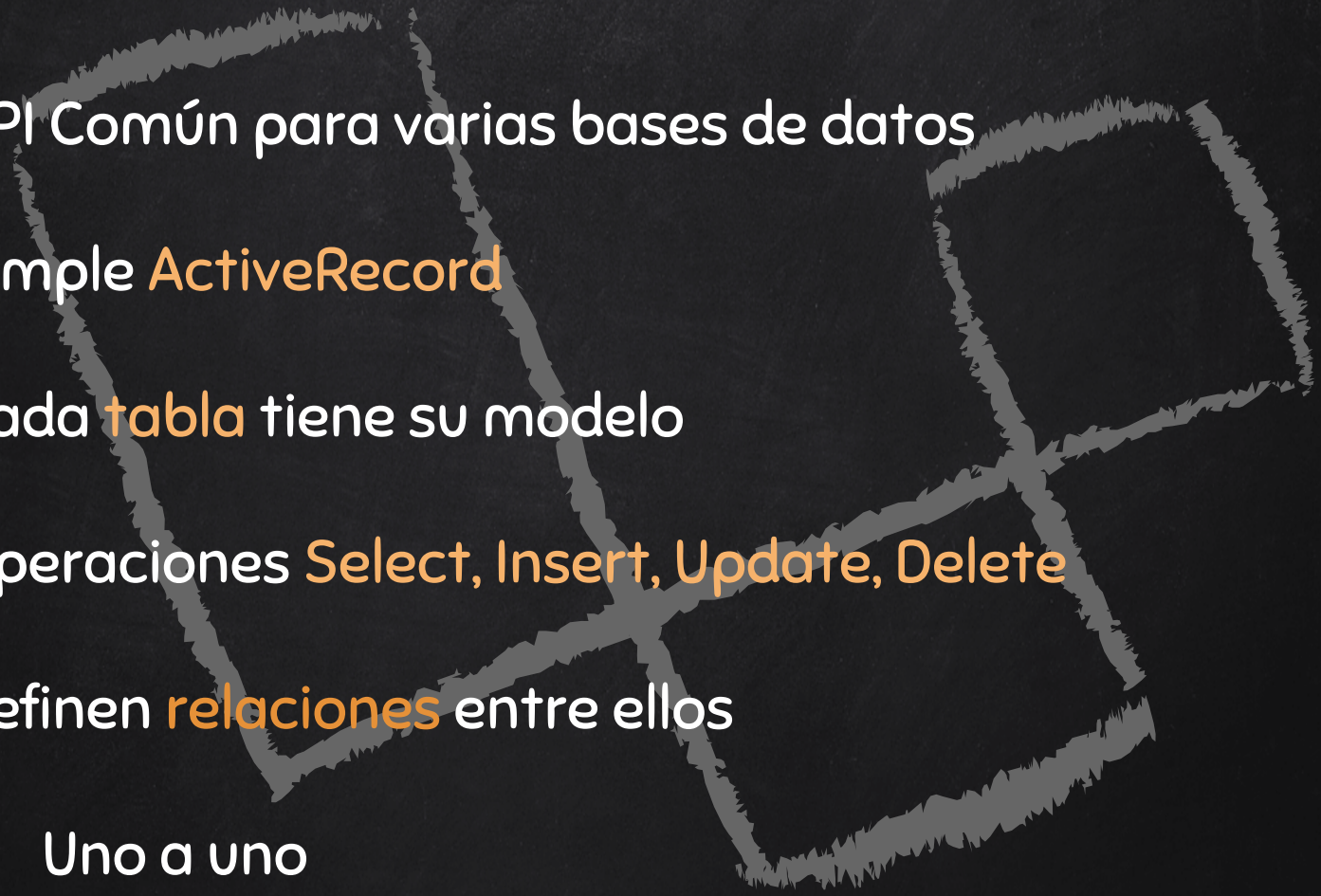


- ✘ "Control de versión" para la base de datos
- ✘ Mapeo de campos de base de datos
- ✘ Fácil de compartir con otros desarrolladores
- ✘ Fácil de hacer cambios al esquema de base de datos
- ✘ Posibilidad de regresar cambios


¡HAGAMOS CÓDIGO!


Modelos y Eloquent

Modelos y Eloquent



- ✘ API Común para varias bases de datos
- ✘ Simple ActiveRecord
- ✘ Cada tabla tiene su modelo
- ✘ Operaciones Select, Insert, Update, Delete
- ✘ Definen relaciones entre ellos
  - Uno a uno
  - Uno a muchos
  - Muchos a muchos


Relaciones


Colecciones

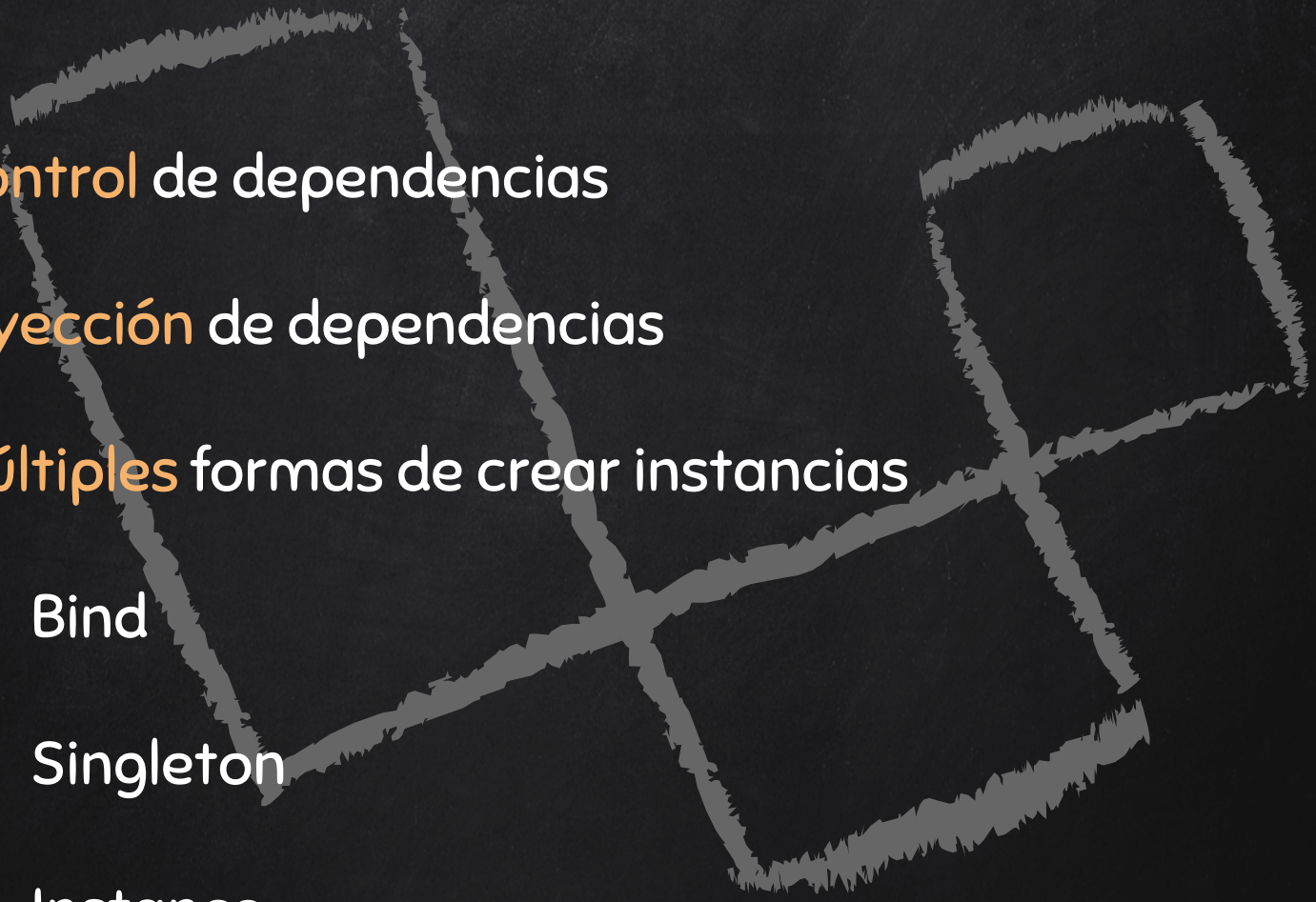

Colecciones


Validación

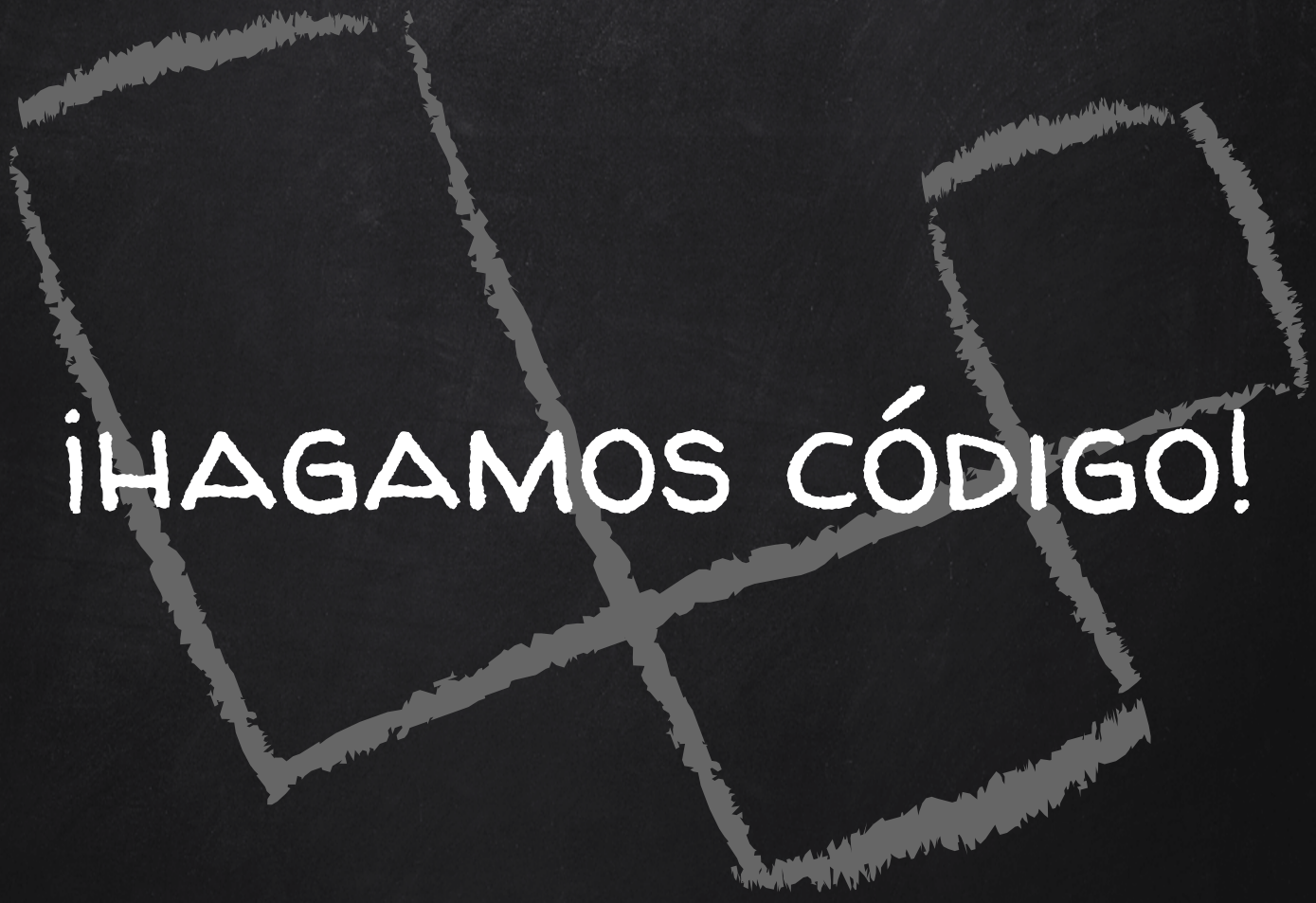

Autenticación


Service Container

Service Container



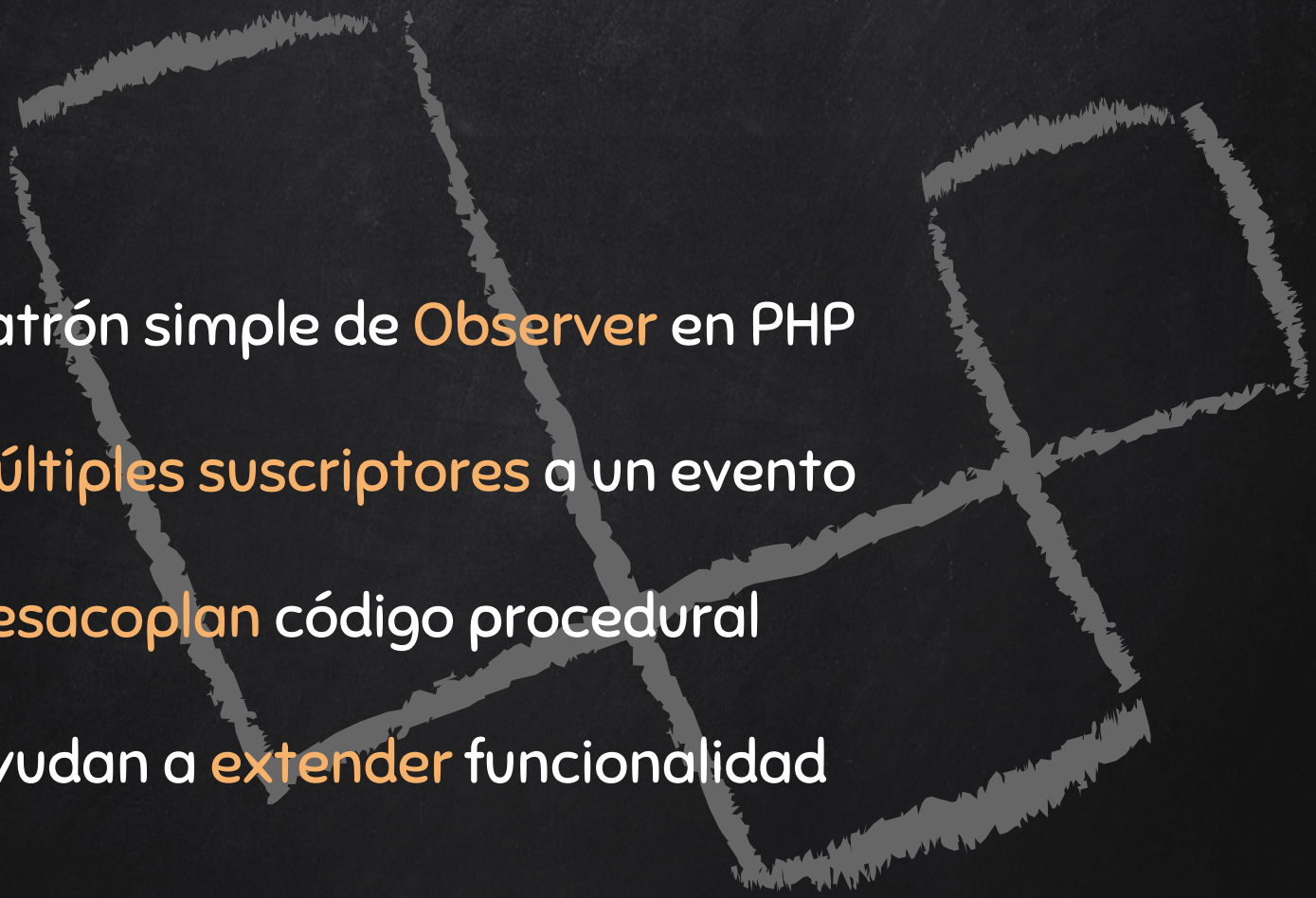
- ✘ Control de dependencias
- ✘ Inyección de dependencias
- ✘ Múltiples formas de crear instancias
  - Bind
  - Singleton
  - Instance


¡HAGAMOS CÓDIGO!

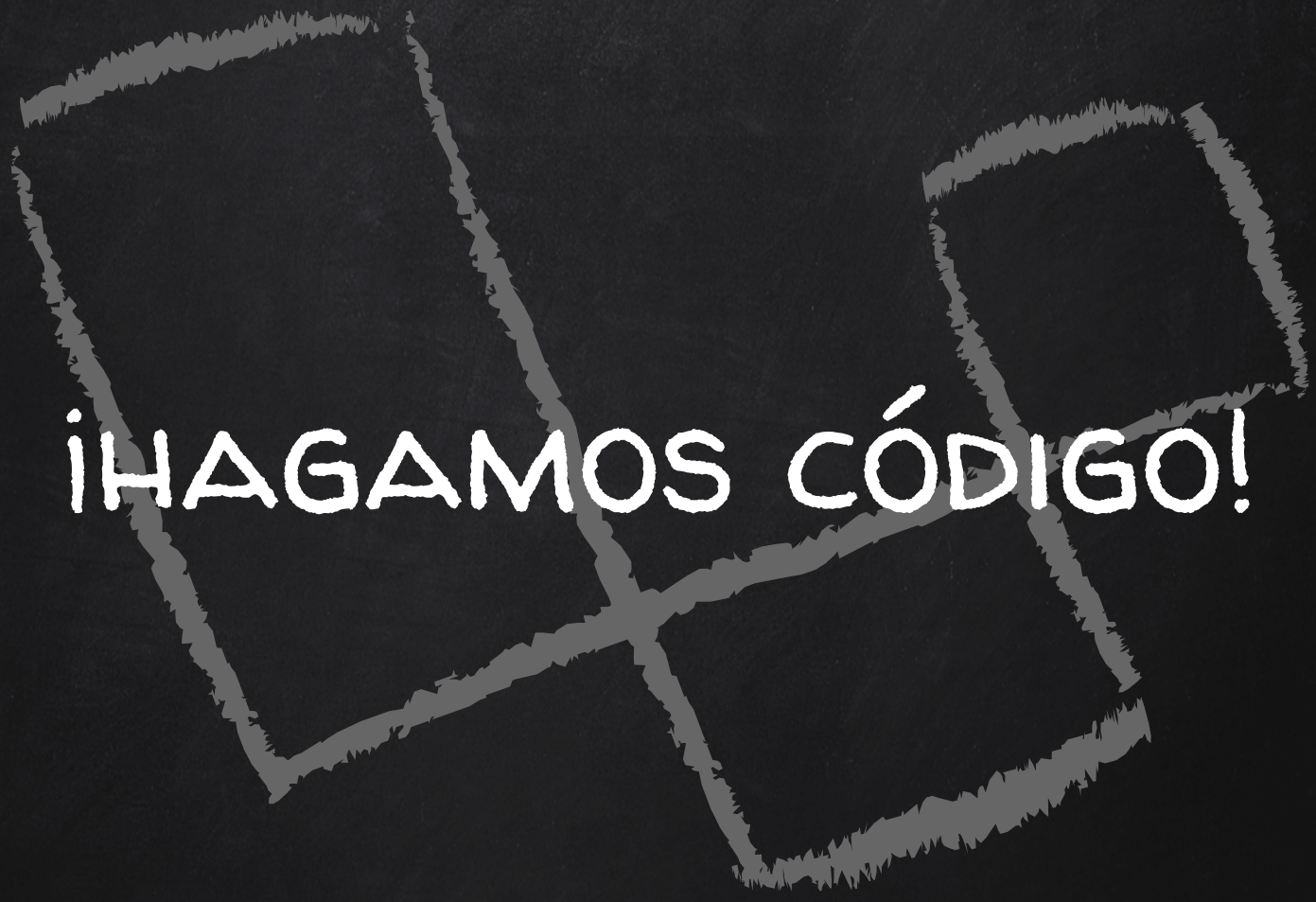

Repositorios

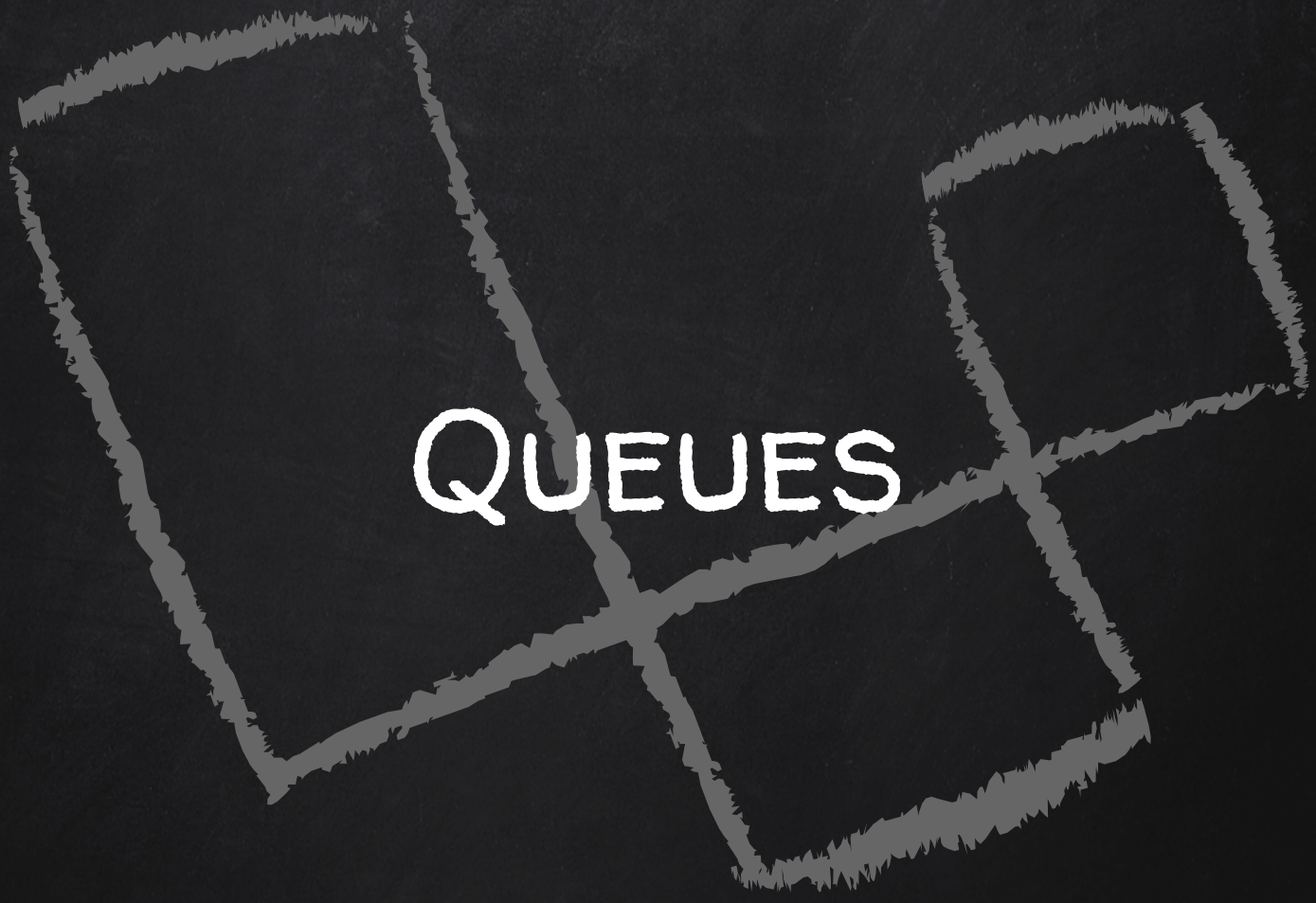

Eventos

Eventos

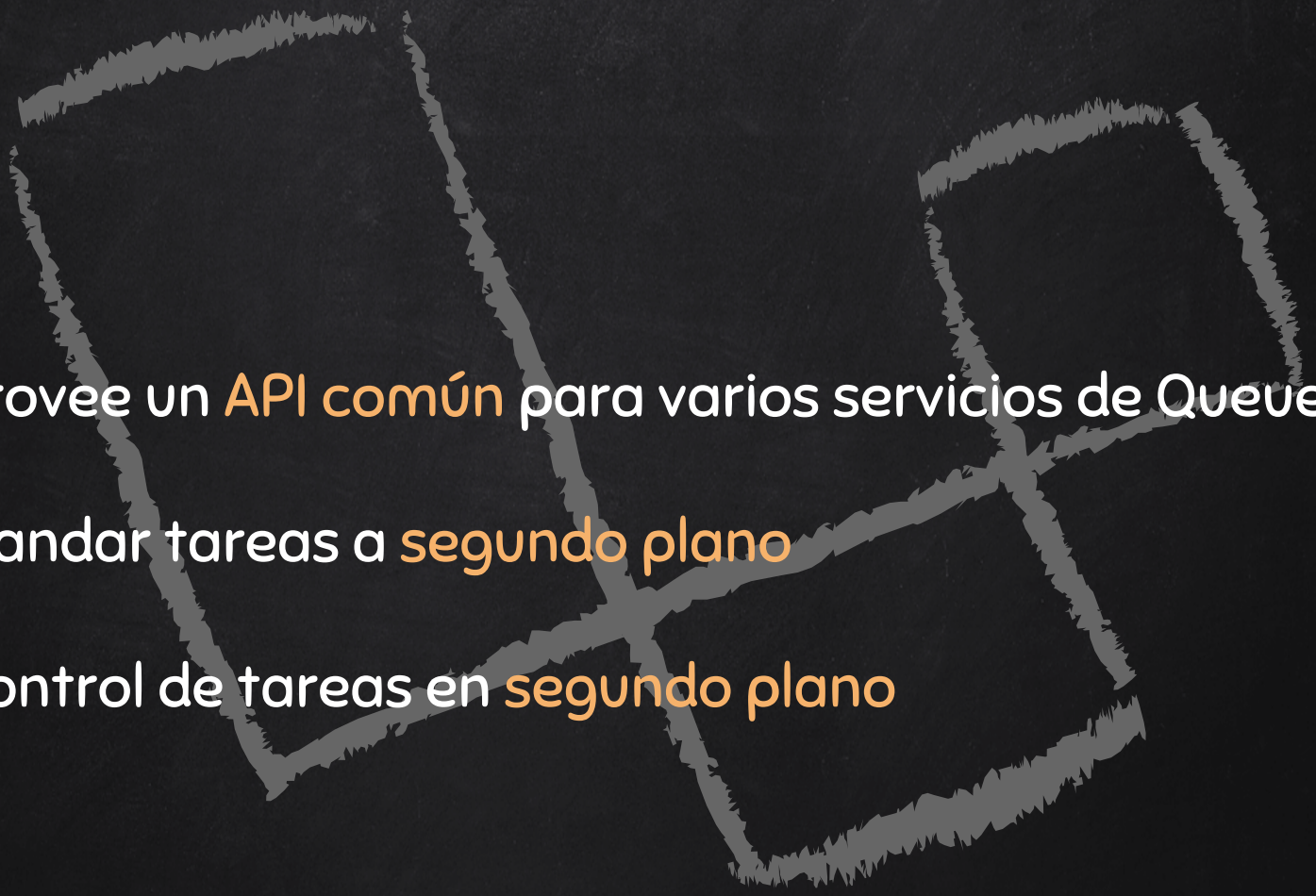


- ✘ Patrón simple de Observer en PHP
- ✘ Múltiples suscriptores a un evento
- ✘ Desacoplan código procedural
- ✘ Ayudan a extender funcionalidad

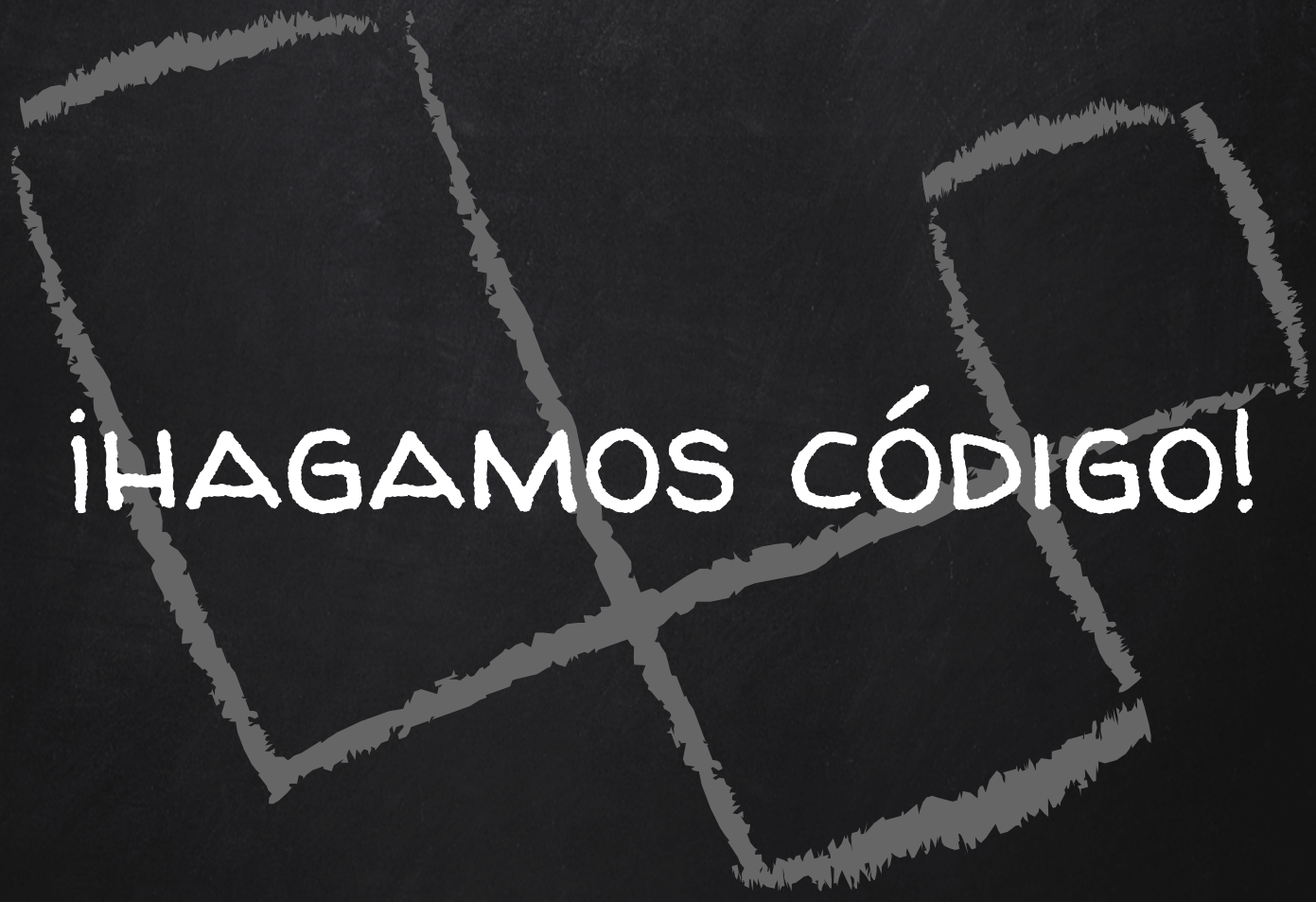

¡HAGAMOS CÓDIGO!

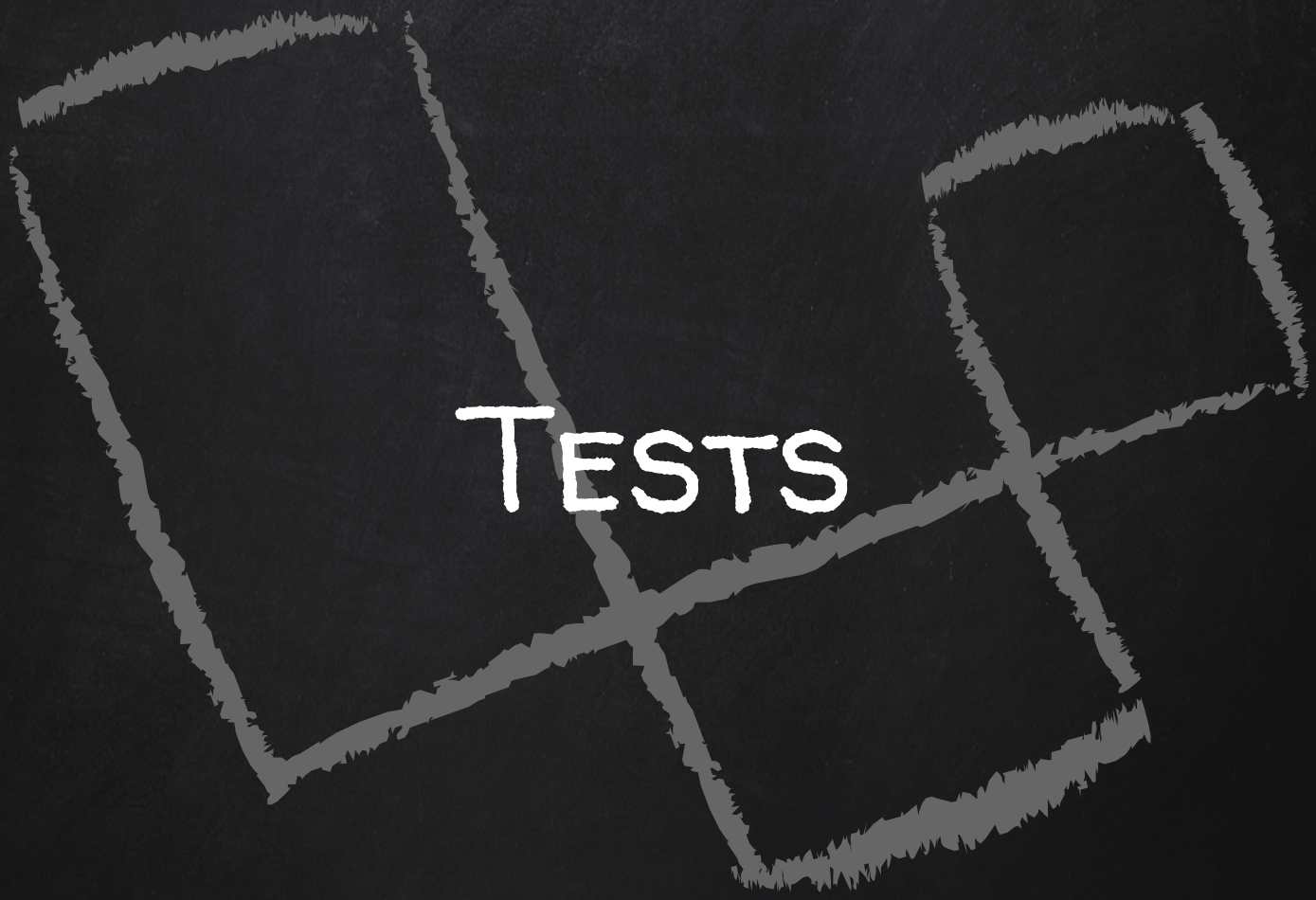

Queues

Queues

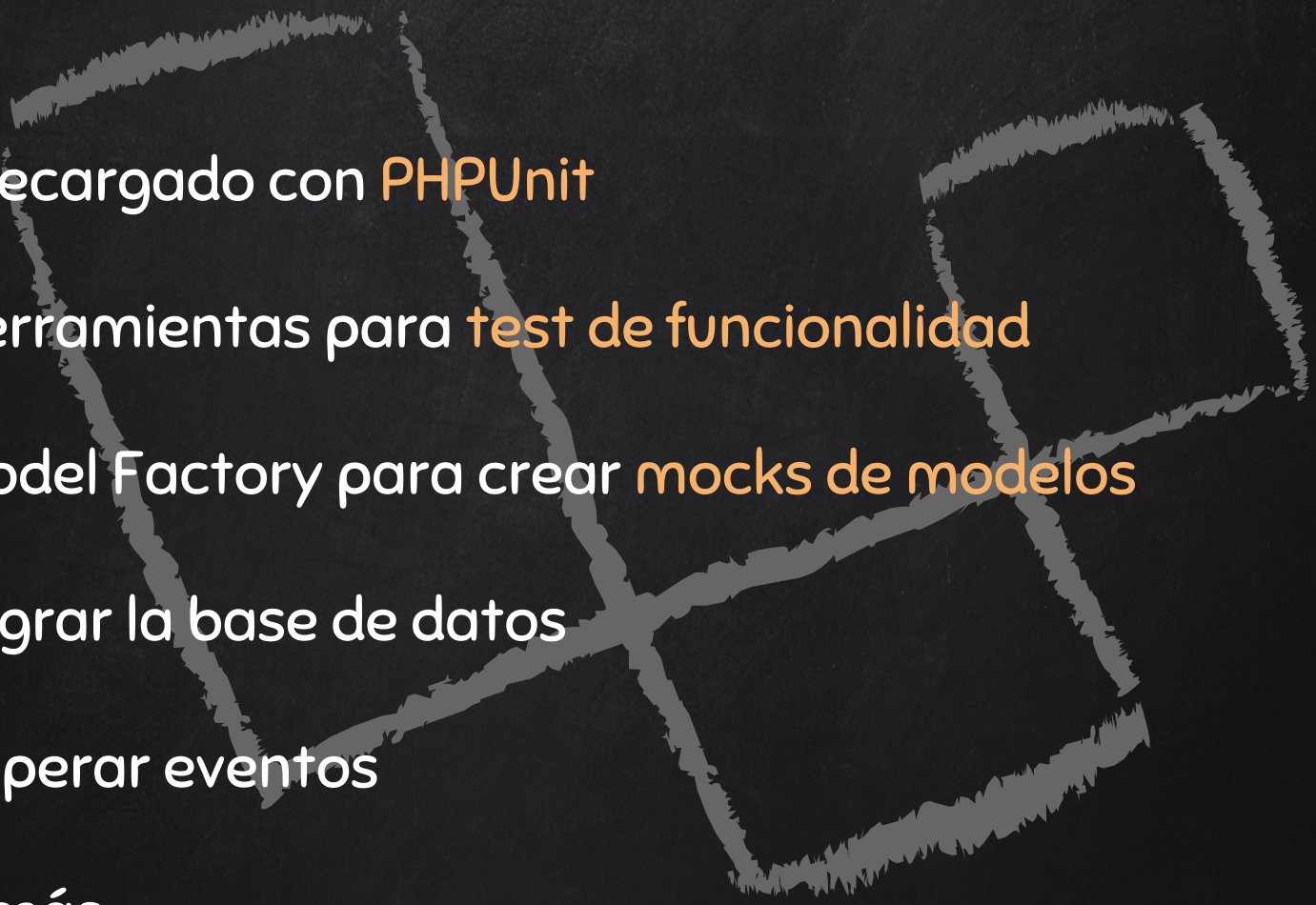


- ✘ Provee un API común para varios servicios de Queues
- ✘ Mandar tareas a segundo plano
- ✘ Control de tareas en segundo plano

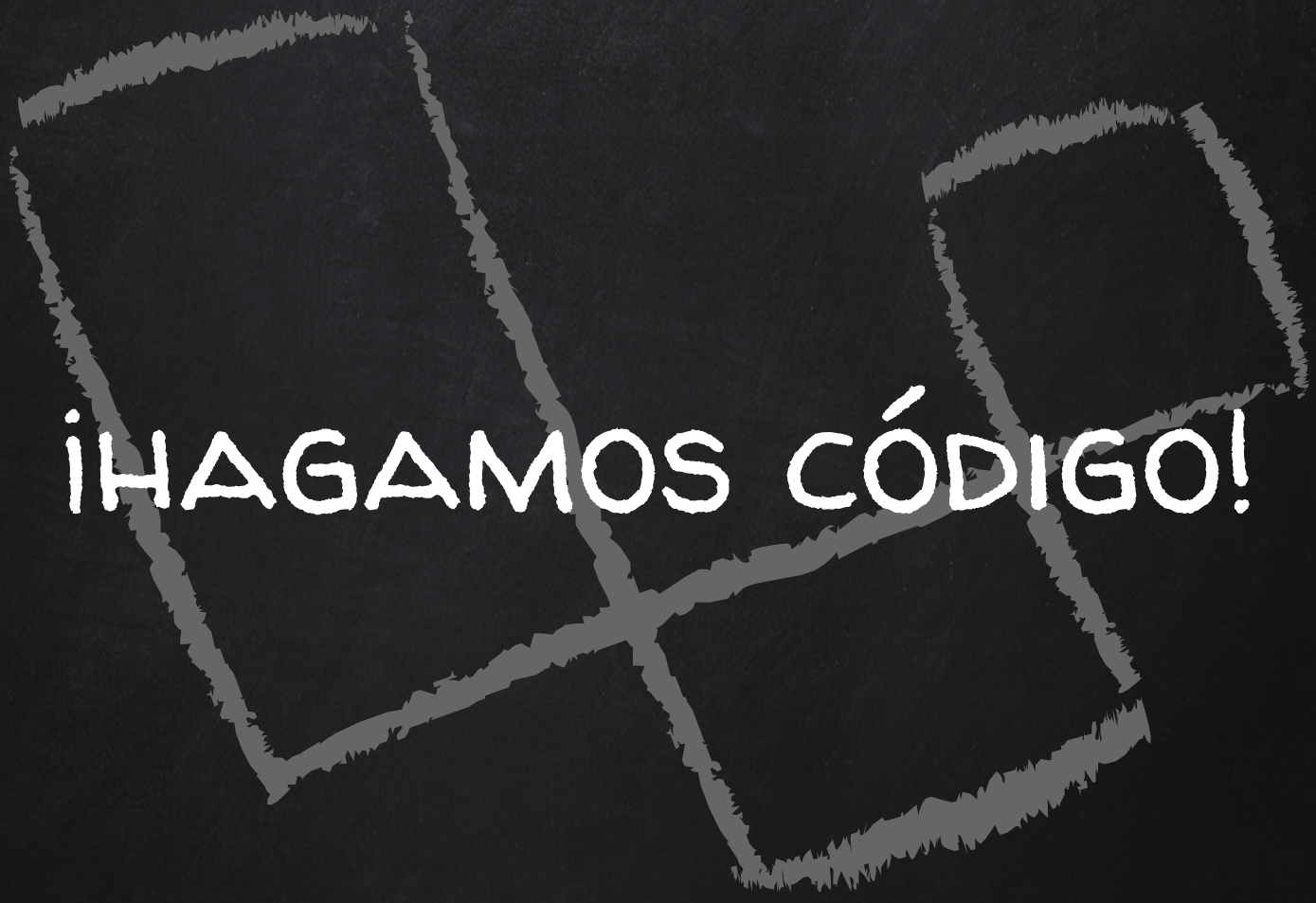

¡HAGAMOS CÓDIGO!


Tests

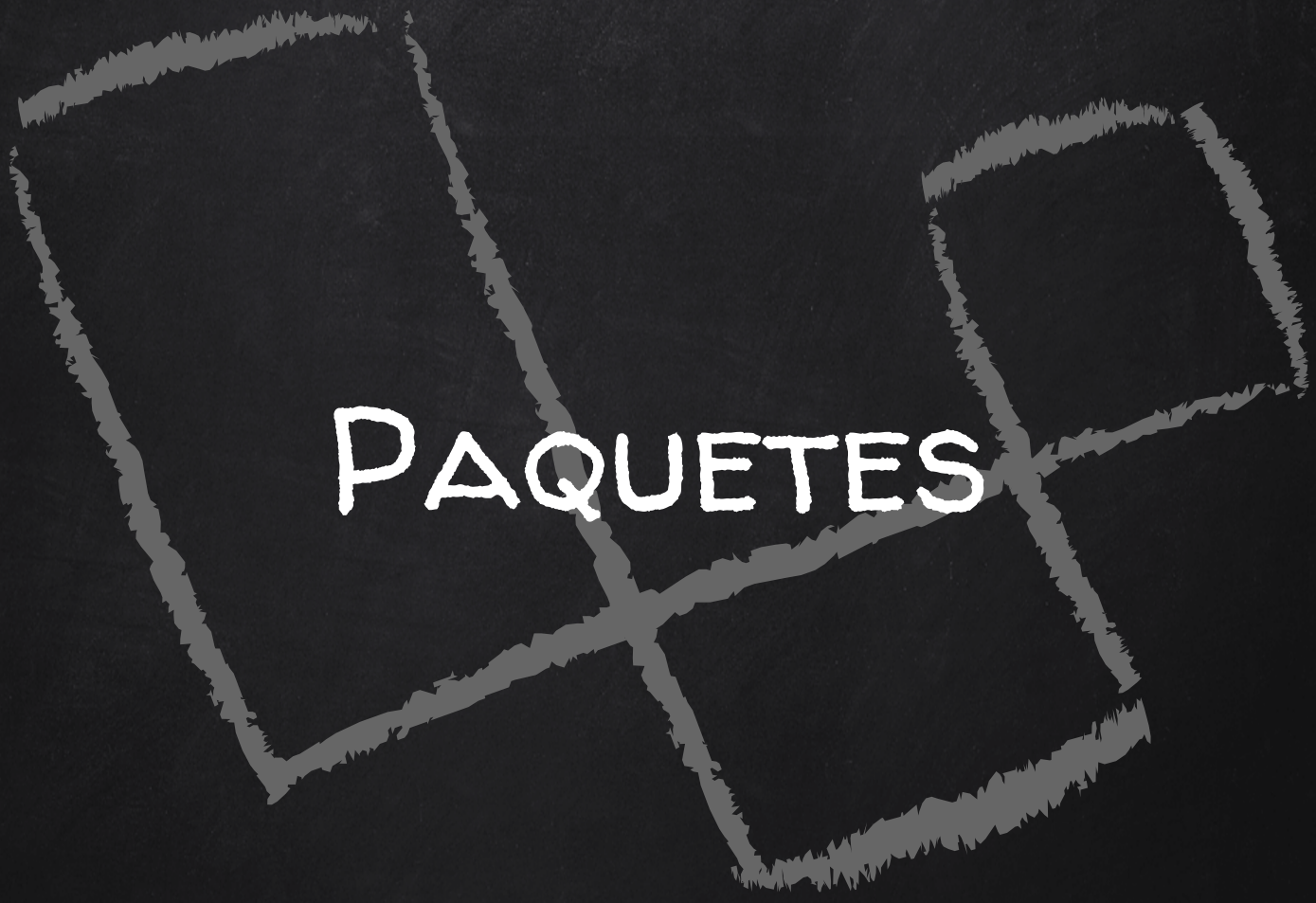
Test

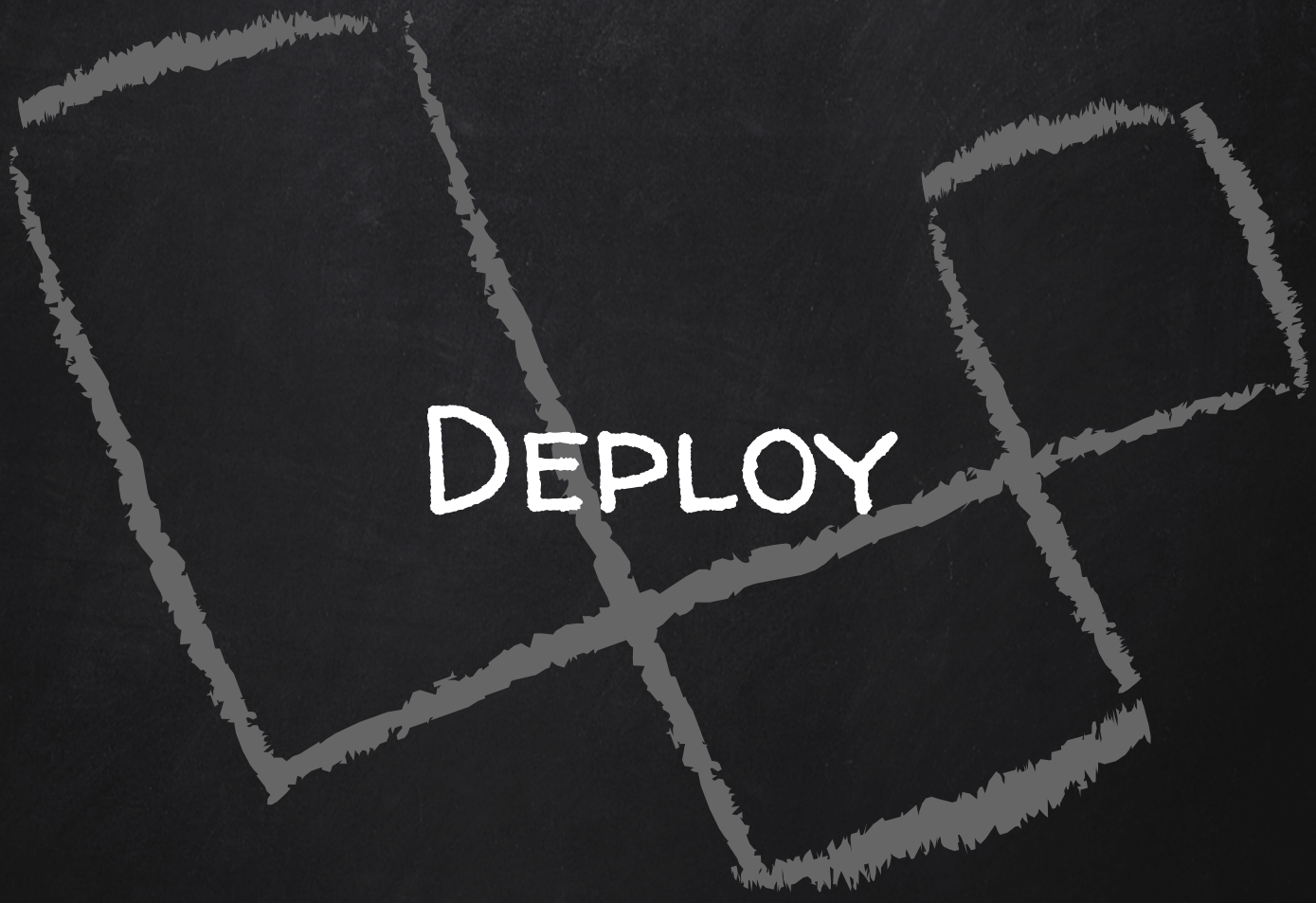


- ✘ Precargado con PHPUnit
- ✘ Herramientas para test de funcionalidad
- ✘ Model Factory para crear mocks de modelos
- ✘ Migrar la base de datos
- ✘ Esperar eventos
- ✘ Y más


¡HAGAMOS CÓDIGO!


Comandos


Paquetes


Deploy

Gracias


@joecohens


/joecohens