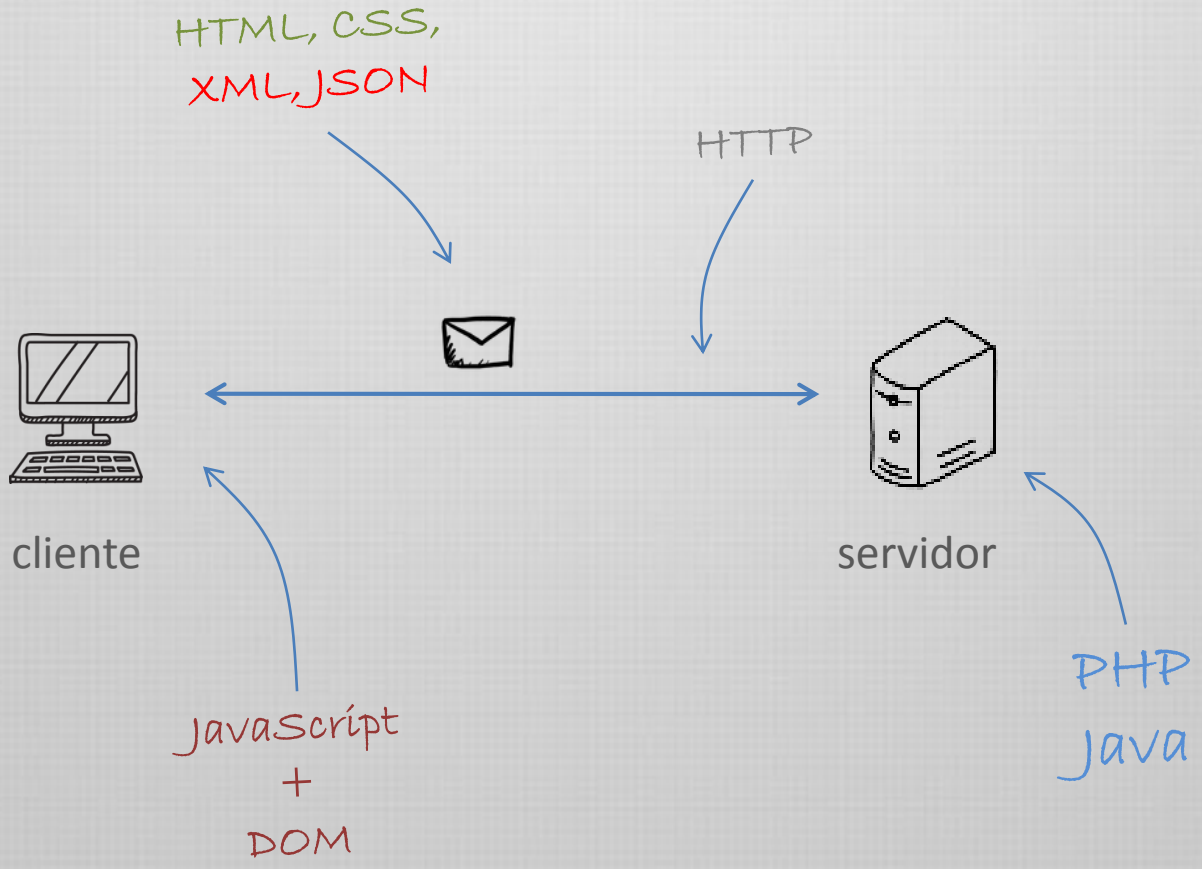
Ingeniería de Aplicaciones Web

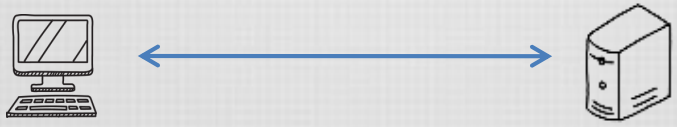
Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Tecnologías web



Arquitecturas Web



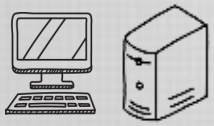
¿cómo organizar todas estas tecnologías?

buscamos

escalabilidad
seguridad
confiabilidad
adaptabilidad

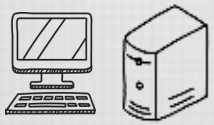
Arquitecturas Web

Arquitecturas web



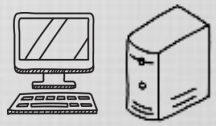
Static HTML

- bajo costo
- simple
- cacheable
- difícil de actualizar
- UI pobre



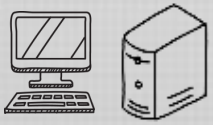
CGI

- Contenido dinámico
- Fácil de actualizar
- Alto costo computacional
- UI pobre



HTML + JavaScript

- Contenido dinámico
- UI Mejorada
- AJAX
- Lógica duplicada
- Riesgo de desbalance
- Poco cacheable



Client Side Apps

- Contenido dinámico
- Reduce carga del servidor
- Cacheable
- Contenido no indexable
- Requiere browser moderno

Frameworks

Básicamente

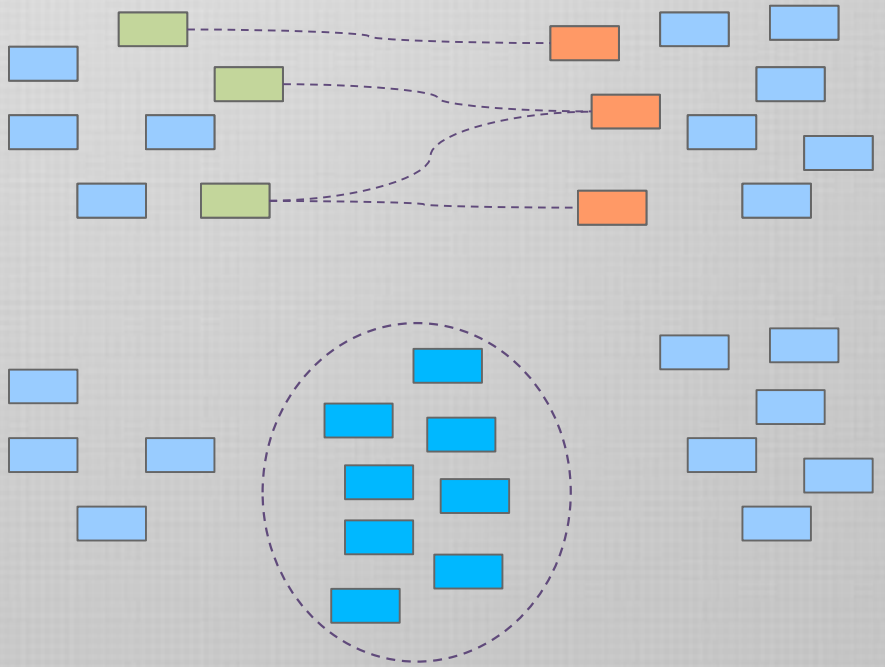
diseño e implementación parcial para una aplicación en un dominio específico

En cierto sentido es una aplicación *incompleta*...

.. y lo faltante puede completarse de diversas formas, de modo tal de obtener diferentes aplicaciones específicas

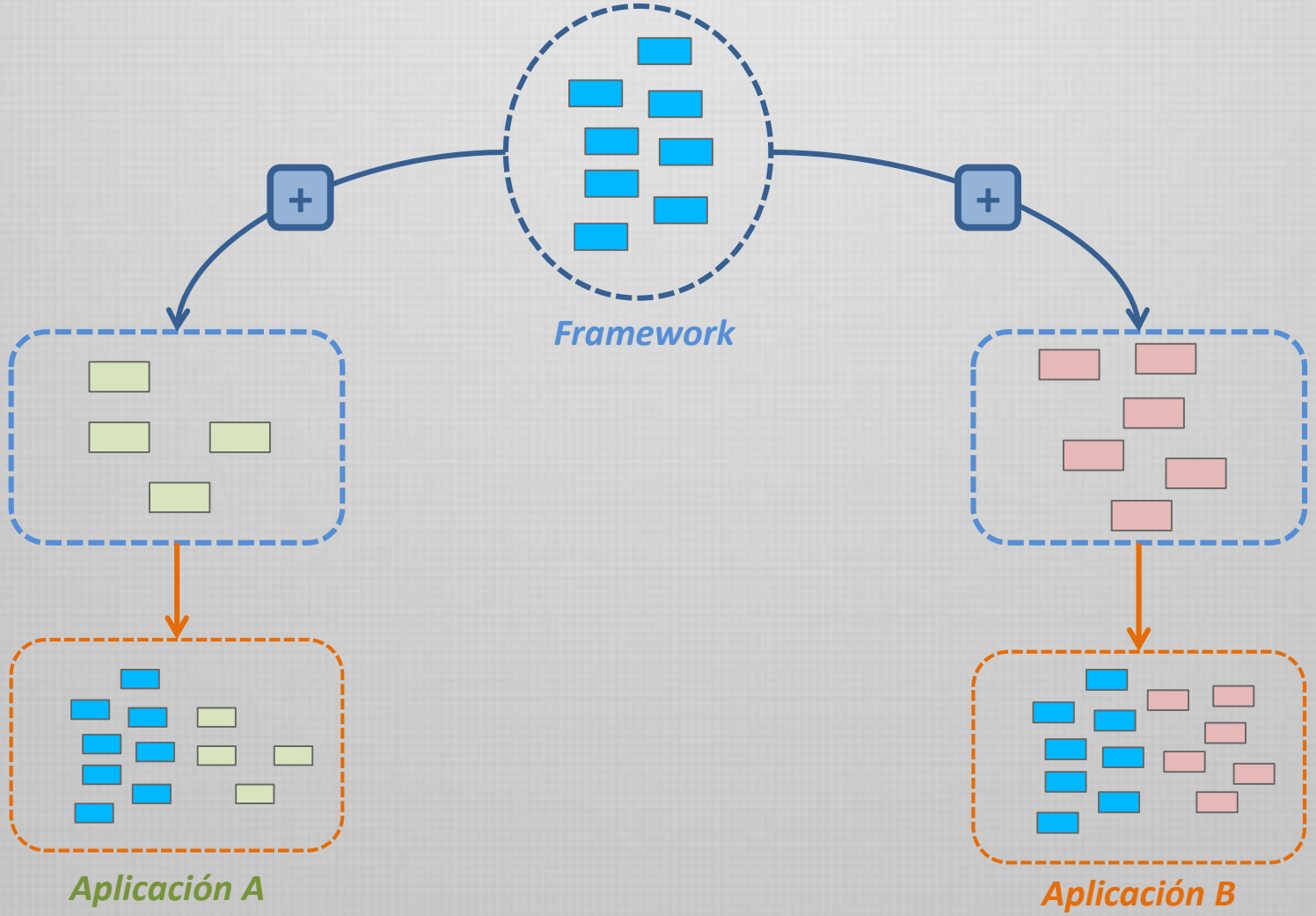
Los frameworks son construídos cuando es necesario desarrollar varias aplicaciones *similares* *(en función y estructura)*

El framework implementa aquellos aspectos comunes a estas aplicaciones.

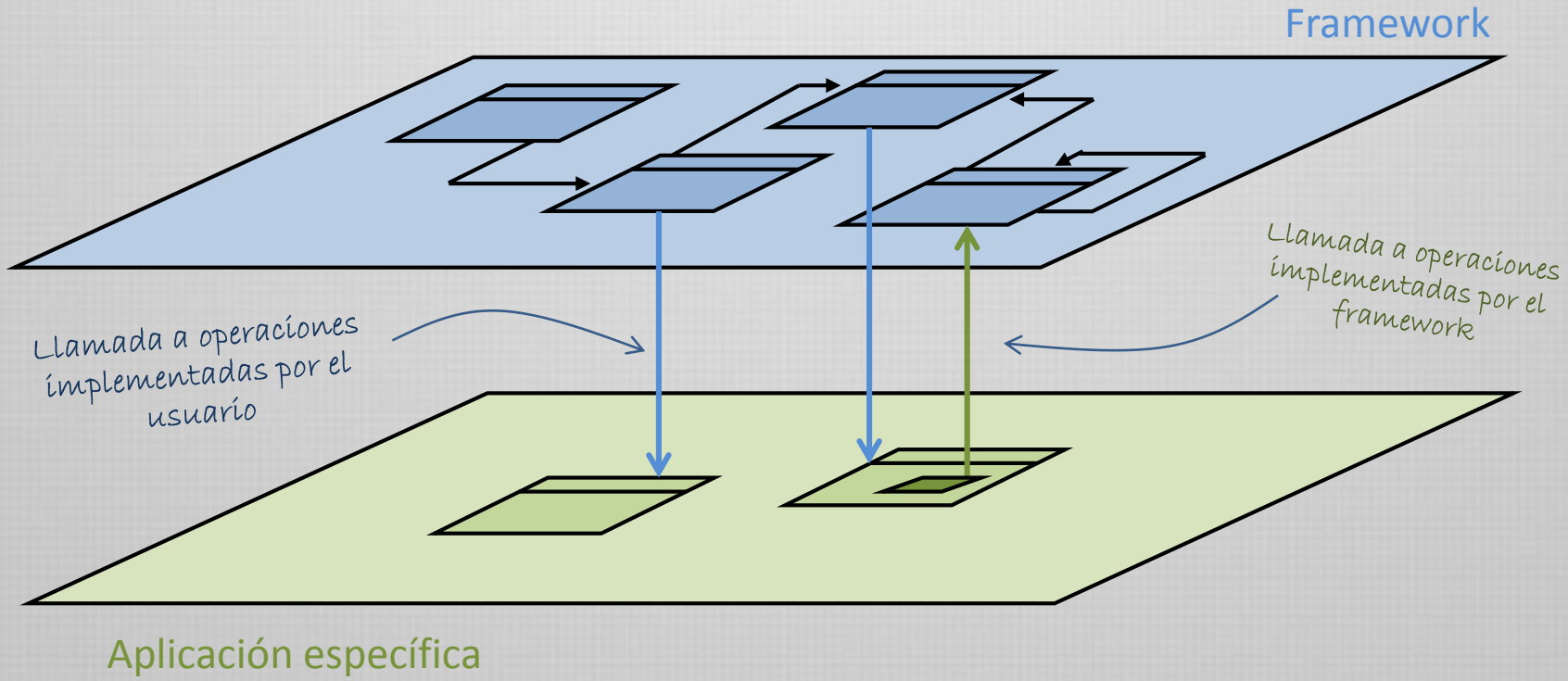


Frameworks

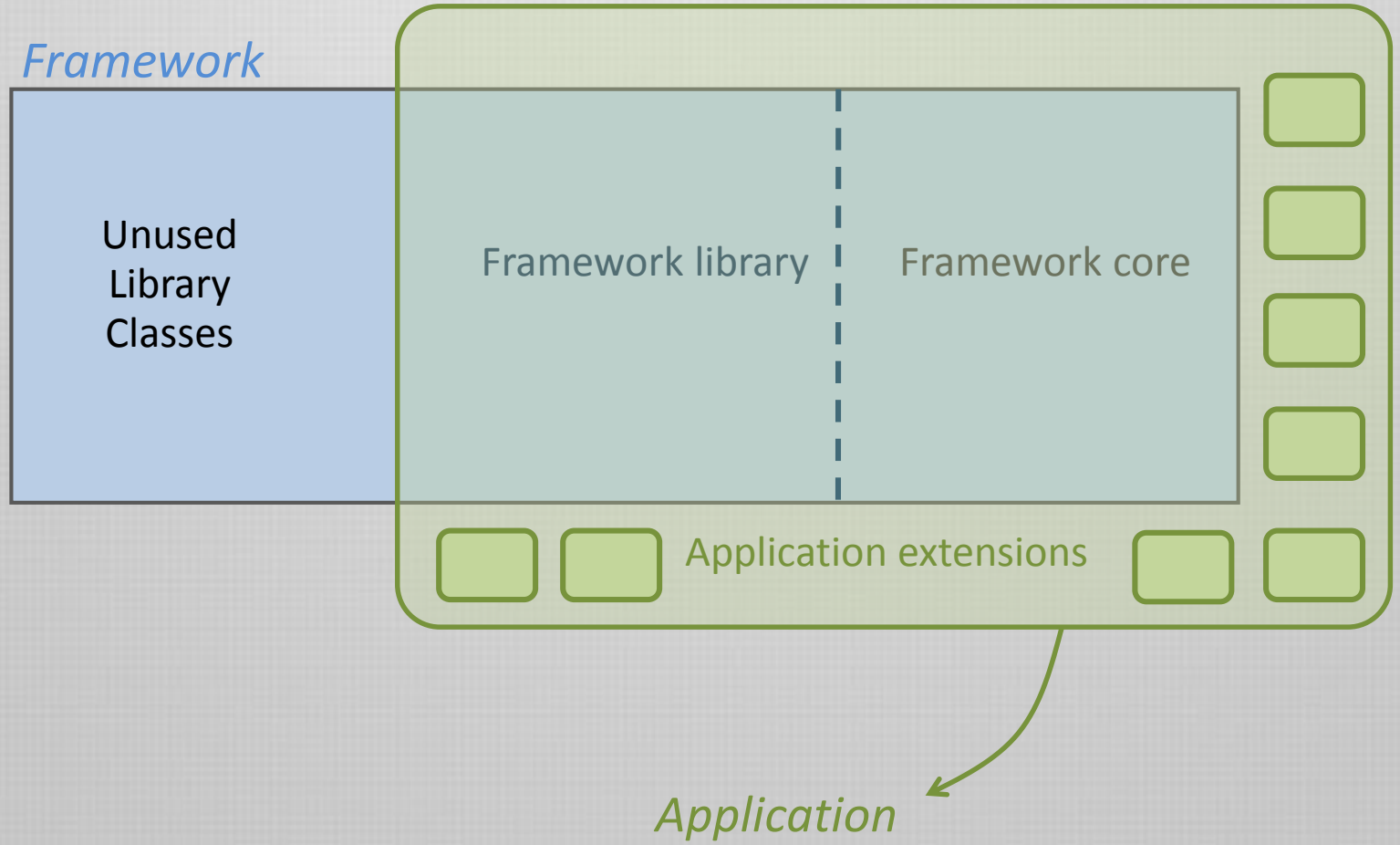
El proceso de obtener una aplicación concreta de un framework se denomina *instanciación del framework* (framework instantiation)



Frameworks – visión conceptual



Partes de un framework



Beneficios del uso de frameworks

Reduce el tiempo de desarrollo.
No se inicia el proyecto desde cero.

Reduce los costos de desarrollo.
Menos tiempo implica menos costos.
El conocimiento del framework implica menos formación de RRHH.

Induce una organización arquitectónica.
Cada framework sigue una arquitectura particular, bastante rígida.
Obliga a una disciplina de trabajo ordenada.

Impregna calidad al proyecto.
Los frameworks son testeados y mantenidos constantemente.
Muchas instanciaciones actualmente en ejecución.

Aumenta el espectro de desarrolladores informados.
Un framework popular es conocido por muchos y por lo tanto ofrece más candidatos desarrolladores.

Facilita la integración y renovación de desarrolladores.
Un desarrollador familiarizado con un framework puede incorporarse rápidamente a proyectos de instanciación.

Layering

La metáfora de las *capas* es la técnica común para particionar sistemas complejos.

Ejemplo mínimo → *cliente-servidor en bases de datos*

Cada capa descansa sobre la capa inferior.

Las capas inferiores son independientes de las superiores.

Ventajas:

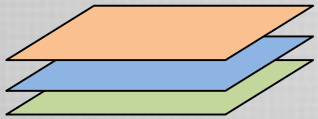
- *Comprensión modular por capas*
- *Abstracción de capas inferiores*
- *Facilidad de cambios de capas*

Desventajas:

- *Las capas no siempre encapsulan todo (cascading changes).*
- *Performance en riesgo*

Las tres capas principales

Las tres capas principales usualmente distinguidas son

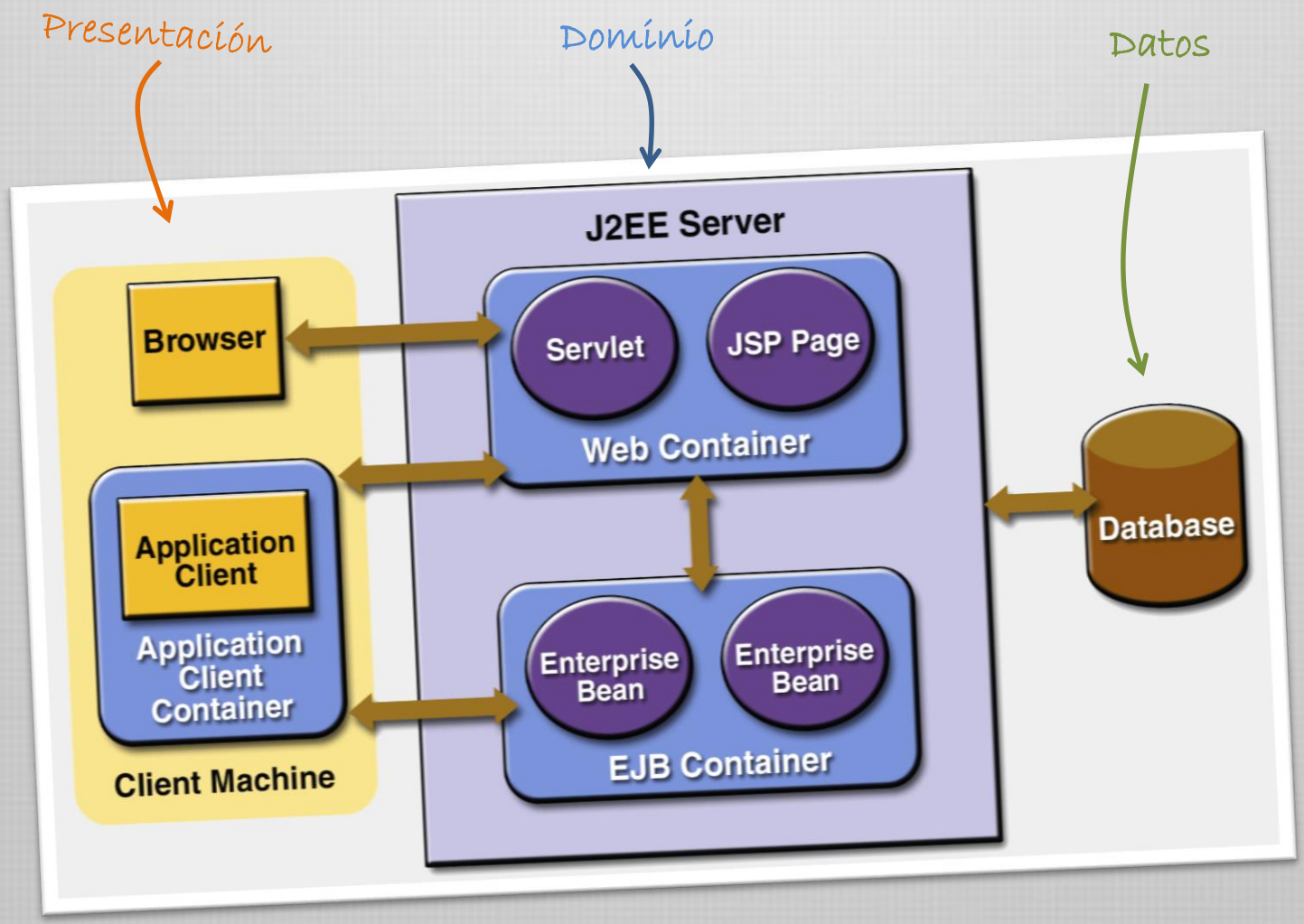


Presentación
Centrada en la interacción con el usuario.
Muestra información e interpreta comandos.

Dominio
"Domain Logic" o "Business Logic"
Computaciones sobre los datos dependientes del dominio particular

Datos
Centrada en la comunicación con otros sistemas externos.
Típicamente un DBMS.

Java 2 Platform



Patrón general de diseño: MVC

MVC es el patrón arquitectónico predominante en las aplicaciones web

El Controlador administra el Modelo y la Vista.

La Vista es responsable de observar el Modelo para exteriorizar los datos.

Patrón general de diseño: MVC

MVC es el patrón arquitectónico predominante en las aplicaciones web

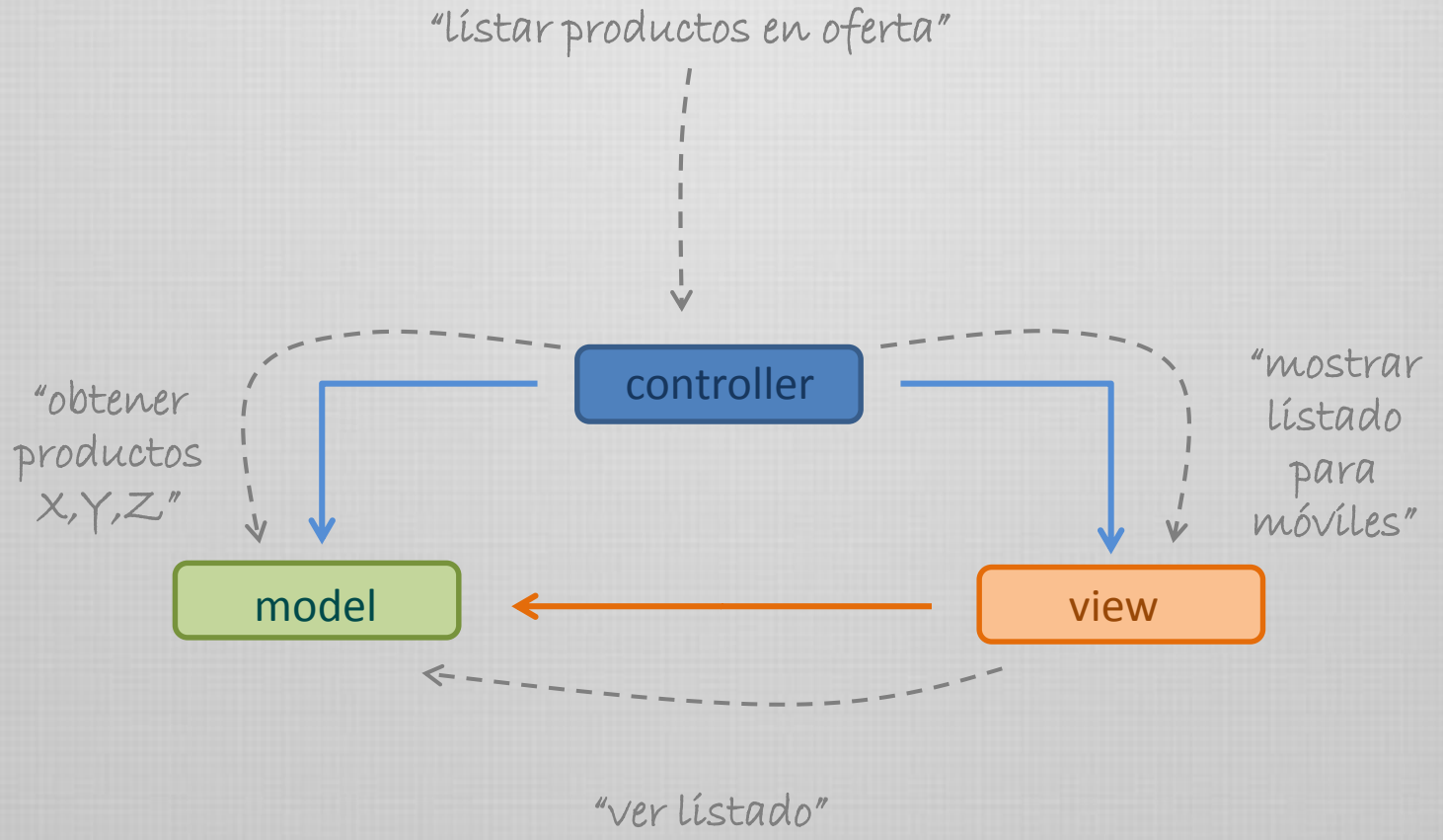
Patrón general de diseño: MVC

controller

model

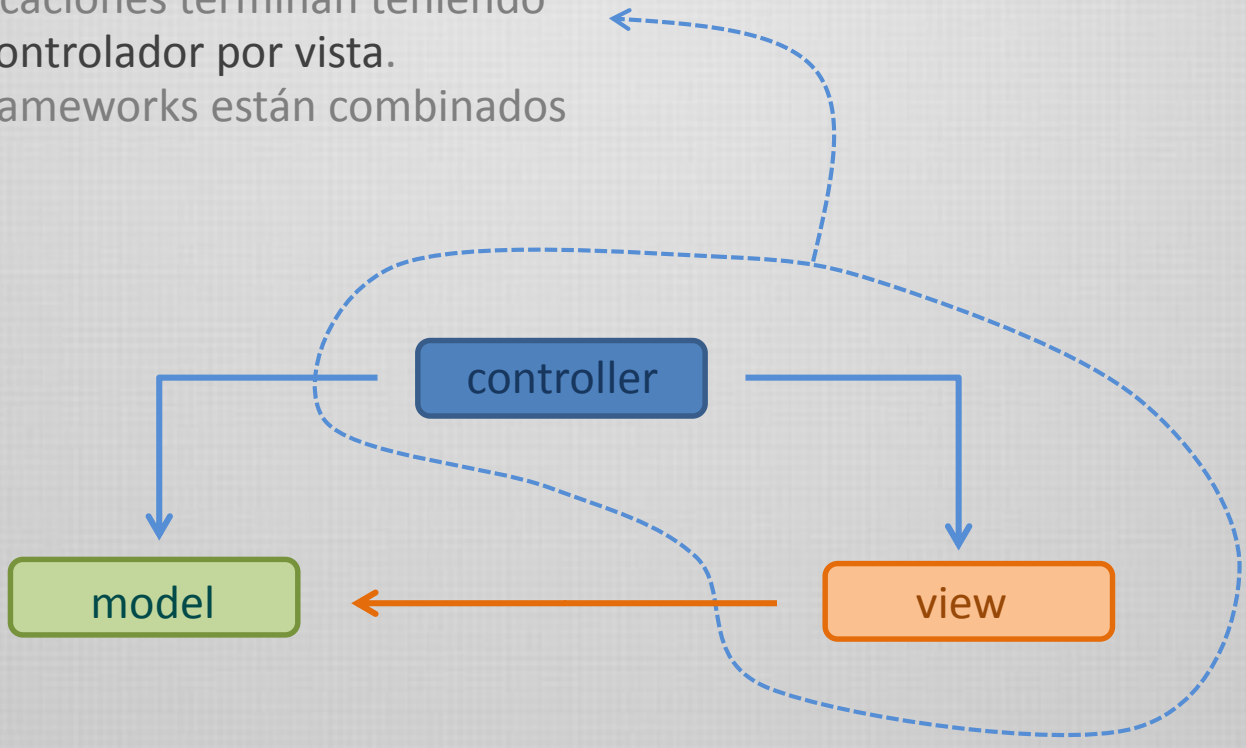
view

Patrón general de diseño: MVC



MVC: separaciones principales

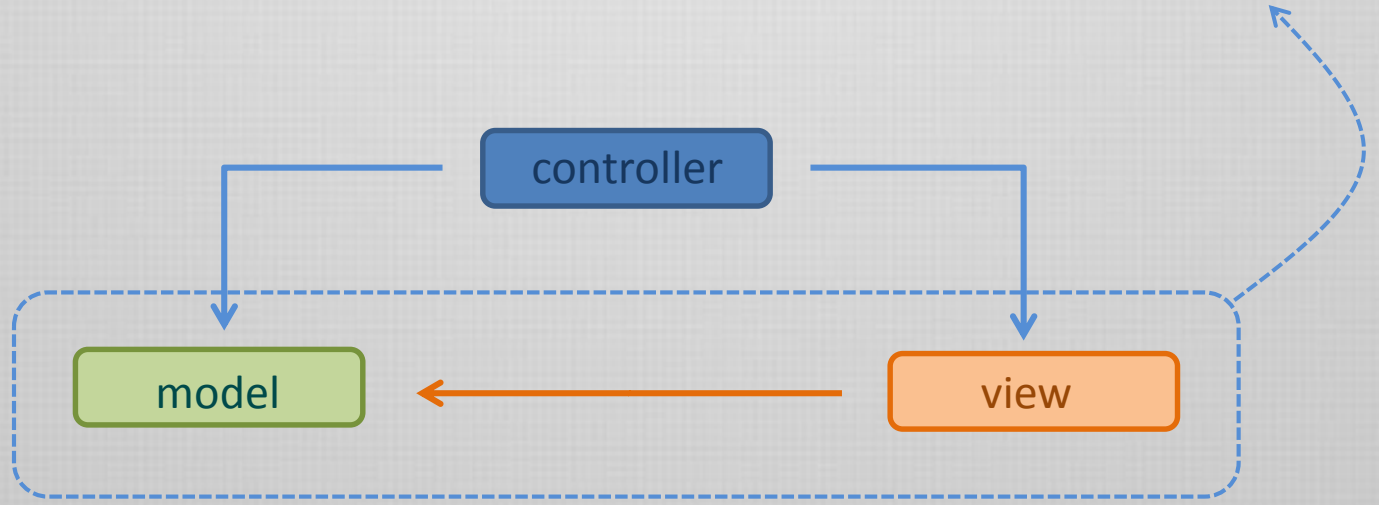
Usualmente sobrevalorada.
Muchas aplicaciones terminan teniendo un controlador por vista.
En algunos frameworks están combinados



MVC: separaciones principales

Separación principal

Las vistas y las presentaciones son problemas puntuales.
Los programadores del modelo no necesitan conocer nada de las vistas.
La misma información puede ser visualizada de maneras diferentes.
El testing puede despojarse de características visuales.

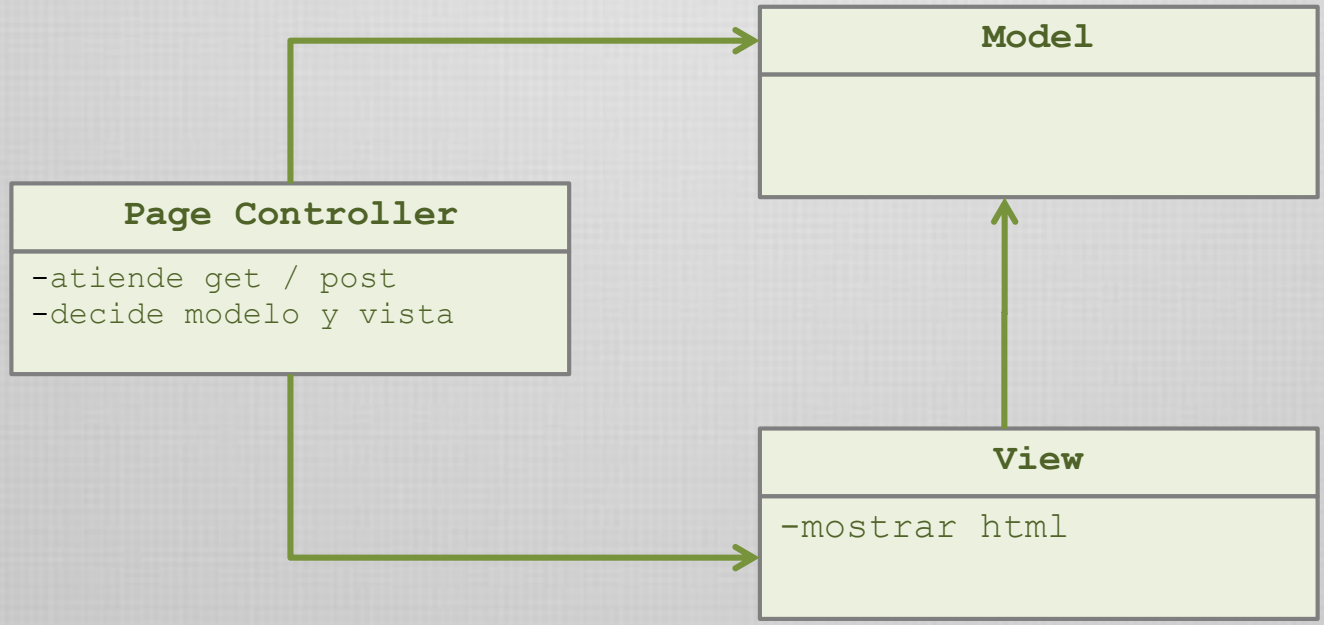


una pagina HTML
un documento PDF
una imagen
datos JSON, XML

Page controller

Page-controller es un patrón de diseño congruente con el patrón arquitectónico MVC.

Aquí un objeto controla el *request* de una página específica del sitio (action)

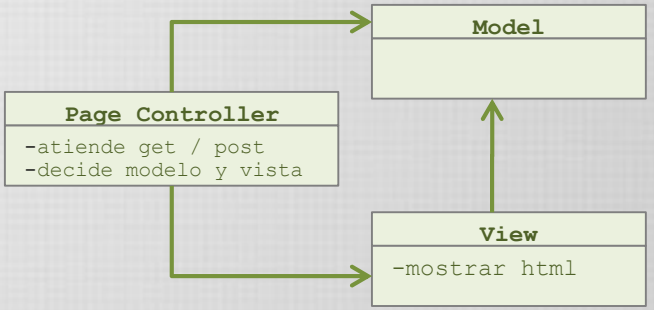


Hay un controlador por cada página lógica del sitio.
En algunos casos es la página misma.

Principalmente vinculado a una acción del sitio
Determina un conjunto de páginas parametrizables.

Page controller

El *page-controller* puede ser
- un *script* (CGI, servlets, etc)
- un *server page* (ASP, PHP, JSP, etc)



Las responsabilidades básicas del *page-controller* son:

- *Decodificar la URL y extraer los datos correspondientes.*

  http://misitio.com/productos/productoA/comprar
- *Crear e invocar los objetos del modelo para procesar los datos.*

  db = new DataBaseConnection(); a = db.getOfertas();
- *Determinar qué vista debe mostrar los resultados y redireccionar.*

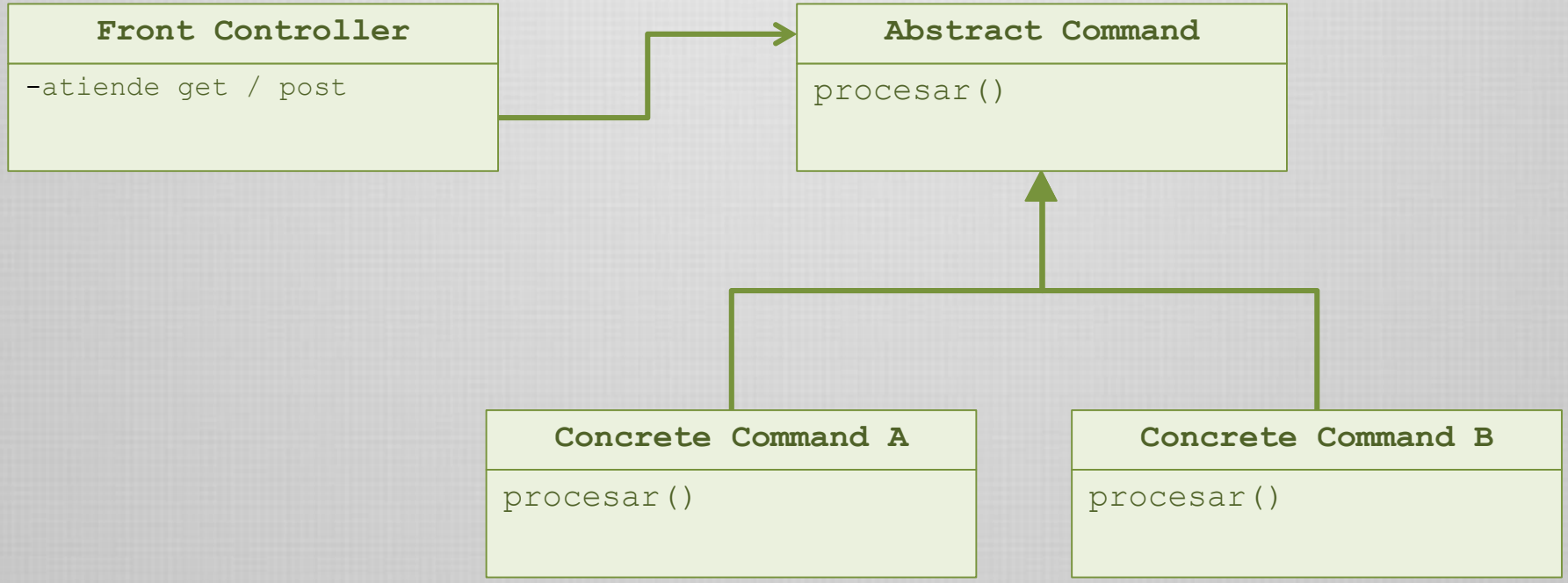
  web, mobile, pdf, word, json, xml..

Al ser un patrón de diseño no necesariamente rige toda la arquitectura del sitio.
Admite incluso la convivencia de los dos modos de *page-controller*.
Es posible manejar algunas URLs como *scripts*, otras como *pages*.

El patrón *page-controller* funciona particularmente bien en sitios de lógica simple.

Front controller

Front-controller maneja *todos* los requests de un sitio web.



El objeto Front-controller recibe el request del cliente y

- Recupera los datos correspondientes.
- Prepara los datos para los comandos.
- Decide qué comando ejecutar a continuación.

Front controller

El objeto *Front-controller* es usualmente implementado como un módulo *script* (php, servlet, asp) en lugar de una *server page*.

El objeto *Front-controller* decide el comando estática o dinámicamente

parseando
URL + lógica condicional.
Simple y fácil de depurar.

parseando
URL + instanciando comandos.
Hace uso de las ventajas del patrón Command.
La clase Command explicitada en el URL, con archivos configurables que mapean el nombre de la clase correspondiente.

Es un patrón más complicado de implementar que Page-controller.
Sólo se implementa **un** Front-controller.

Template View

Template View es un patrón que renderiza información a HTML, incrustando marcas en una plantilla HTML.

```
{ marca }
<? marca ?>
<pre:marca/>
<prefijo:marca> </prefijo:marca>
```

Las marcas son reemplazadas por datos computados.
Es en realidad el patrón habitual de los scripts PHP, JSP, ASP, aunque estos permiten la inclusión de lógica compleja.

Es el patrón usualmente elegido para la Vista del patrón MVC