Perfecto, he leído el material del **Tema 1: Arquitecturas y Herramientas de Programación Web** que me compartiste. En lugar de limitarme a resumirlo, voy a reorganizar y reexplicar los conceptos de manera más clara, con ejemplos y comparaciones, para que te sirva como apuntes bien estructurados y fáciles de estudiar para el examen.

# **📘 Tema 1 – Arquitecturas y Herramientas de Programación Web**

## **1. Introducción**

El desarrollo web en entorno servidor forma parte del **modelo cliente-servidor**, que es la base de cómo funciona internet. Para entender este tema debemos tener claros tres ejes:

1. **Cómo se comunican cliente y servidor** (arquitecturas).
2. **Qué lenguajes y tecnologías existen** para programar aplicaciones en el servidor.
3. **Qué herramientas se utilizan** para desarrollar y gestionar estos proyectos.

## **2. Modelo Cliente/Servidor**

La web funciona gracias a una interacción:

* El **cliente** (normalmente un navegador) solicita información.
* El **servidor** recibe la petición, la procesa y envía una respuesta.

Ejemplo: cuando entras en Gmail e introduces usuario y contraseña, el servidor valida tus datos y construye dinámicamente la página de tu bandeja de entrada.

### **🔑 Capas en el modelo cliente-servidor**

La arquitectura suele dividirse en tres capas:

1. **Capa de presentación (cliente):**
   * Es la parte visible para el usuario (interfaz).
   * Suele estar hecha con HTML, CSS y JavaScript.
   * Se encarga de mostrar la información y recoger las acciones del usuario.
2. **Capa de negocio (lógica):**
   * Contiene las reglas y procesos que hacen funcionar la aplicación.
   * Ejemplo: verificar que un usuario existe antes de mostrar su correo.
   * Puede ejecutarse en el cliente o en el servidor, aunque lo habitual es en el servidor.
3. **Capa de datos (persistencia):**
   * Gestiona dónde se guardan y consultan los datos (bases de datos).
   * Normalmente en el servidor, con sistemas como MySQL, MariaDB o PostgreSQL.

### **📌 Tipos de arquitecturas**

* **Fat Client (Thin Server):** el cliente hace casi todo el trabajo. Ejemplo: una aplicación que descarga muchos datos y los procesa en tu PC.
* **Fat Server (Thin Client):** el servidor hace todo el trabajo y el cliente solo muestra la información. Ejemplo: Gmail.
* **Dos capas:** cliente y servidor de base de datos.
* **Tres capas:** cliente, servidor de aplicaciones (lógica) y servidor de base de datos.

## **3. Páginas Web: Estáticas vs Dinámicas**

* **Estáticas:**
  + Siempre muestran lo mismo.
  + Se guardan en el servidor en HTML tal cual.
  + Ejemplo: la página de contacto de una empresa con dirección y teléfono.
* **Dinámicas:**
  + Se generan en el momento en función de la petición del usuario.
  + El servidor ejecuta un programa (PHP, Python, etc.) que construye el HTML a medida.
  + Ejemplo: tu bandeja de entrada en Gmail.

👉 **Diferencia clave:** Las estáticas son fijas, mientras que las dinámicas cambian según el usuario, sus acciones o los datos disponibles.

## **4. Lenguajes de Programación en el Servidor**

### **4.1 Lenguajes de Scripting**

Son lenguajes que se mezclan con HTML y son interpretados por el servidor.  
 Ejemplos:

* **PHP:** gratuito, muy popular, orientado a objetos, funciona en casi todos los servidores.
* **ASP (Microsoft):** tecnología antigua sustituida por ASP.NET.
* **Perl:** pionero en desarrollo web, muy usado en CGI.
* **Python:** sencillo, potente, multiplataforma. Base de frameworks como Django o Flask.
* **JSP (Java Server Pages):** combina Java con HTML, convirtiendo el código en Servlets para mejorar el rendimiento.

### **4.2 Aplicaciones CGI**

* **CGI (Common Gateway Interface):** permite que el servidor ejecute un programa externo (en C, Perl, etc.) y devuelva su resultado al navegador.
* Problema: poco eficiente, porque cada petición crea un nuevo proceso.
* Soluciones: **FastCGI** (un solo proceso atiende varias peticiones) o servidores Java con **Servlets** y **JavaBeans**.

### **4.3 Aplicaciones Híbridas (ASP.NET)**

* Combinan características de scripting y de programas compilados.
* ASP.NET permite usar C#, VB.NET o JScript.NET.
* Ofrece gran rendimiento y está totalmente orientado a objetos.
* Desventaja: es una plataforma propietaria (Microsoft).

### **4.4 Comparativa**

* **Scripting:** fácil y rápido de modificar, pero menos eficiente.
* **CGI/Compilados:** más rápidos, pero menos flexibles y difíciles de integrar.
* **Híbridos:** buen equilibrio entre rendimiento y portabilidad.

## **5. Integración con Servidores Web**

El servidor web (Apache, Nginx, IIS...) puede:

* Servir contenido estático (HTML, imágenes).
* Ejecutar contenido dinámico (PHP, JSP, ASP.NET).
* Aceptar peticiones con distintos métodos:  
  + **GET:** se envían datos en la URL (ejemplo: ...?id=5).
  + **POST:** se envían datos ocultos (formularios de login).

## **6. Herramientas de Programación**

### **6.1 Editores de Texto**

Permiten escribir código con resaltado de sintaxis y utilidades básicas:

* Notepad++, Sublime Text, Atom, Visual Studio Code.

### **6.2 Entornos de Desarrollo (IDE)**

Más completos, con depuración, compilación y pruebas:

* Eclipse, NetBeans, Visual Studio, Dreamweaver.

### **6.3 Herramientas adicionales**

* **Tratamiento de imágenes:** Photoshop, GIMP.
* **Bases de datos:** MySQL Workbench, phpMyAdmin.

## **7. Servicios Web para Desarrollo**

### **7.1 XAMPP**

* Paquete que incluye Apache (servidor web), MariaDB/MySQL (base de datos) y PHP/Perl.
* Multiplataforma, gratuito y de código abierto.
* Ideal para montar un servidor local en tu PC.

### **7.2 MySQL Workbench**

* Herramienta visual para diseñar, administrar y consultar bases de datos MySQL.
* Alternativa más avanzada a phpMyAdmin.

# **📌 Conclusión**

Este tema sienta las bases del desarrollo web en entorno servidor:

* Entender **cómo se organizan cliente y servidor**.
* Diferenciar entre **páginas estáticas y dinámicas**.
* Conocer **los principales lenguajes del servidor** y sus ventajas/inconvenientes.
* Aprender a usar **herramientas de desarrollo** y configurar un entorno con **XAMPP + MySQL Workbench**.

¿Quieres que te lo organice también en un **esquema visual tipo mapa conceptual** (ideal para repaso rápido antes del examen) además de esta explicación detallada?