Desarrollo Web en Entorno Servidor

Ejercicios Tema 1

James Edward Nuñez Cuzcano

2ºDaw 2025/26

Indice de ejercicios

1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS.....4

2. Modelo de comunicaciones cliente – servidor y su relación con las aplicaciones web.....4

3. Estudio sobre los métodos de petición HTTP /HTTPS más utilizados.....5

4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme)/URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.....6

5. Modelo de desarrollo de aplicaciones multicapa – comunicación entre capas – componentes – funcionalidad de cada capa.....6

6. Modelo de división funcional front-end / back-end para aplicaciones web.....7

7. Página web estática – página web dinámica – aplicación web – mashup 8

8. Componentes de una aplicación web.....8

9. Programas ejecutados en el lado del cliente y programas ejecutados en el lado del servidor - lenguajes de programación utilizados en cada caso.....9

10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual).....10

11. Características y posibilidades de desarrollo de una plataforma XAMPP.....12

12. En qué casos es necesaria la instalación de la máquina virtual Java (JVM) y el software JDK en el entorno de desarrollo y en el entrono de explotación.....14

13. IDE más utilizados (características y grado de implantación actual).....14

14. Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual).....14

15. Apache HTTP vs Apache Tomcat.....14

16. Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual).....14

17. Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, ….....14

18. Repositorios de software – sistemas de control de versiones: GIT , CVS, Subversion, ...14

19. Propuesta de configuración del entorno de desarrollo para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USED y xxx-WXED.....14

20. Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE.....14

21. Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web:.....14

CMS – Sistema de gestión de contenidos.....14

ERP – Sistema de planificación de los recursos empresariales.....14

22. Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:.....15

• MEAN (con MongoDB y con MySQL) • Java EE vs Spring • Microsoft .NET • Angular 7 • Symfony • Laravel • CakePHP • CodeIgniter.....15

1. Protocolos de comunicaciones: IP, TCP, HTTP, HTTPS.

El protocolo IP (Internet Protocol) es un protocolo de comunicación de la capa de red, que se encarga de transmitir datos en forma de paquetes bidireccionalmente entre el origen y destino

El protocolo TCP (Transmission Control Protocol) es un protocolo de transmisión de datos de la capa de transporte que sirve para crear conexiones dentro de una red para intercambiar datos. Este protocolo garantiza que los datos lleguen al destinatario íntegros y en el mismo orden que se mandaron

El protocolo HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación que permite la transferencia de información en forma de archivos.

El protocolo HTTPS (Protocolo Segura de Transferencia de Hypertexto) es un protocolo de la capa de aplicación, utiliza un cifrado basado en ssl y tsl para crear un canal cifrado para la transferencia segura de hypertexto, en resumen es la versión segura de http.

2. Modelo de comunicaciones cliente – servidor y su relación con las aplicaciones web.

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta, esta es la arquitectura fundamental para el desarrollo web.

3. Estudio sobre los métodos de petición HTTP /HTTPS más utilizados.

-HEAD: se utiliza para solicitar que el servidor solo envíe el encabezado de la respuesta, sin el archivo.

-GET: solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

-POST: se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

-PUT: se utiliza para actualizar un recurso existente, o para crearlo en caso de que no exista todavía.

-DELETE: borra un recurso en específico.

-CONNECT: establece un túnel hacia el servidor identificado por el recurso.

-OPTIONS: se utiliza para comprobar que opciones de comunicación posee el recurso de destino.

-TRACE: realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

-PATCH: es utilizado para aplicar modificaciones parciales a un recurso.

4. Estudio sobre el concepto de URI (Identificador de Recursos Uniforme)/URL/URN, estructura, utilidad y relación con el protocolo HTTP/HTTPS.

La URL (Localizador Uniforme de Recursos) indica la ubicación del recurso y el protocolo utilizado para acceder a él. Incluye información como el dominio, la ruta, y puede contener otros elementos como el puerto, parámetros de consulta y fragmentos. Es única en el contexto de la web, y su objetivo principal es localizar recursos accesibles a través de un protocolo.

La URN (Nombre Uniforme de Recursos) identifica un recurso de manera única, pero no indica cómo acceder a él ni el protocolo que se usa. La URN solo da un nombre único al recurso, pero no tiene información sobre su localización.

La URI (Identificador Uniforme de Recursos) es el término genérico que abarca tanto a las URLs como a las URNs. Una URI puede ser una URL, cuando se especifica cómo acceder al recurso, o una URN, cuando solo se indica su nombre sin importar cómo acceder a él.

5. Modelo de desarrollo de aplicaciones multicapa – comunicación entre capas – componentes – funcionalidad de cada capa.

-Capa de presentación: Es la interfaz que interactúa directamente con el usuario, mostrando información y capturando sus acciones. Envía las solicitudes de usuario a la capa de negocio para su procesamiento.

Sus componentes suelen ser

-Capa de lógica de negocio: Procesa y aplica las reglas y la lógica del negocio sobre los datos recibidos. Actúa como intermediaria entre la presentación y el acceso a datos.

-Capa de acceso a datos: Se encarga de acceder, almacenar y recuperar información en la base de datos. Oculta los detalles técnicos de la persistencia y ofrece datos a la capa de negocio.

6. Modelo de división funcional front-end / back-end para aplicaciones web.

El front-end gestiona la interfaz y la experiencia del usuario; el back-end maneja la lógica, datos y procesos del servidor.

En una aplicación web, la división funcional entre front-end y back-end permite separar responsabilidades:

Front-end (lado del cliente):

Es la parte visible de la aplicación. Incluye el diseño de la interfaz, la experiencia de usuario, la interacción y la lógica ejecutada en el navegador. Se desarrolla con HTML, CSS y JavaScript (y frameworks como React, Angular o Vue).

Back-end (lado del servidor):

Es la parte oculta para el usuario. Gestiona la lógica de negocio, el procesamiento de peticiones, la autenticación, la seguridad y el acceso a bases de datos. Se construye con lenguajes como Java, Python, Node.js, PHP, Ruby y se apoya en sistemas de bases de datos (MySQL, PostgreSQL, MongoDB, etc.).

La comunicación entre ambos se realiza mediante API (REST o GraphQL) que permite el intercambio de datos en formato JSON o XML.

Este modelo aporta modularidad, escalabilidad y mantenimiento más sencillo en el desarrollo de aplicaciones web.

7. Página web estática – página web dinámica – aplicación web – mashup .

Una página web estática es un conjunto de archivos (HTML/CSS) que muestra contenido fijo creado por el desarrollador; una página web dinámica genera su contenido en el servidor o en el cliente según la petición o los datos (usuario, base de datos), por lo que cambia en cada visita; una aplicación web es un sistema interactivo y completo que se comporta como software en el navegador, con lógica de negocio, estado y comunicación continua entre front-end y back-end; y un mashup combina y presenta datos o servicios de múltiples fuentes externas (APIs, feeds, mapas) en una sola interfaz. La diferencia principal reside en cómo y cuándo se genera el contenido (precreado vs generado bajo demanda), el nivel de interactividad y lógica (informativa vs funcional y con estado) y la dependencia de fuentes externas (mashups integran terceros mientras las otras pueden ser autónomas).

8. Componentes de una aplicación web.

Una aplicación web simple suele tener estos componentes esenciales: front-end (interfaz), back-end (servidor/ lógica), API (comunicación), base de datos (persistencia) y hosting/CDN (despliegue y entrega) — cada uno con una función clara: mostrar, procesar, comunicar, guardar y servir.

Front-end (cliente): la interfaz que ve y usa el usuario en el navegador o en un webview. Se encarga de mostrar datos y capturar acciones (HTML/CSS/JavaScript o frameworks como React/Vue).

Back-end (servidor): procesa las peticiones, aplica la lógica de la aplicación (reglas, validaciones) y prepara los datos para enviar al cliente. Puede ser una única aplicación en Node/Python/Java, etc.

API: puntos de comunicación entre front y back (por ejemplo endpoints REST o GraphQL). Define qué datos se piden y cómo se responden.

Base de datos: almacenamiento persistente de la información (usuarios, contenidos, pedidos). Puede ser relacional (Postgres/MySQL) o NoSQL (MongoDB) según necesidad.

Hosting / CDN: donde se publica la app y desde donde se sirven los archivos. El hosting ejecuta el servidor; la CDN entrega rápidamente los recursos estáticos (JS, CSS, imágenes).

9. Programas ejecutados en el lado del cliente y programas ejecutados en el lado del servidor - lenguajes de programación utilizados en cada caso.

Los programas del lado del cliente corren en el navegador o dispositivo del usuario y gestionan la interfaz, interacción y parte de la lógica local (principalmente con HTML/CSS/JavaScript/TypeScript y WebAssembly); los programas del lado del servidor corren en un servidor y gestionan la lógica de negocio, datos y procesos seguros (usando Node.js/JavaScript, Python, Java, C#, PHP, Ruby, Go, Rust, Kotlin, etc.). La diferencia clave es dónde se ejecutan (cliente vs servidor), qué confianza/seguridad se requiere y qué recursos (acceso a DB, CPU prolongada) pueden usar.

-Lado del cliente: renderizan la interfaz, gestionan eventos del usuario, validan formularios en el navegador, reproducen multimedia y usan APIs del navegador.

Lenguajes / tecnologías comunes:

-HTML & CSS: estructura y presentación.

-JavaScript / TypeScript (React, Vue, Angular, Svelte): lógica, interacción

-WebAssembly (WASM) — código compilado desde Rust, C/C++, Go para cómputo intensivo en el navegador.

-Lado del servidor : reciben y procesan peticiones, aplican lógica de negocio, acceden y modifican bases de datos, gestionan autenticación/autorización, realizan procesamiento intensivo o en segundo plano, generan contenido, exponen APIs y orquestan integraciones con terceros.

Lenguajes / plataformas comunes:
-JavaScript / TypeScript (Node.js, Deno): muy usado full-stack.
-Python (Django, Flask, FastAPI): popular por rapidez de desarrollo.
-Java / Kotlin (Spring, Ktor): empresas y aplicaciones con alto rendimiento/estabilidad.
-C# (.NET): aplicaciones empresariales.
-PHP (Laravel, Symfony): muchos sitios web tradicionales.
-Ruby (Rails): desarrollo rápido de MVPs.
-Go: servicios concurrentes y microservicios.
-Rust: servicios de alto rendimiento y seguridad.

10. Lenguajes de programación utilizados en el lado servidor de una aplicación web (características y grado de implantación actual).

JavaScript / TypeScript (Node.js)
Características: event-driven, non-blocking I/O, gran eco de paquetes (npm). TypeScript añade tipado estático y mejores herramientas.
Grado de implantación: Muy alto en startups y proyectos full-stack; extensivo en microservicios y APIs. JavaScript/TS aparecen entre los lenguajes más usados en encuestas de desarrolladores.

Python
Características: sintaxis clara, acelerado desarrollo, fuerte en scripting, APIs y ML; excelentes frameworks web (Django, Flask, FastAPI).
Grado de implantación: Elevado y en crecimiento (fuerte presencia en data/AI, Python ha escalado en popularidad en GitHub y métricas generales.

Java
Características: maduro, JVM, fuerte tipado, rendimiento estable, amplio ecosistema empresarial. Ideal para sistemas transaccionales y grandes plataformas.
Grado de implantación: Muy alto en empresas medianas/grandes y sistemas críticos; sigue siendo pilar en el mundo enterprise.

C# (.NET)
Características: lenguaje moderno en .NET, buen rendimiento, integración con Windows/Azure, productivo para APIs y servicios.
Grado de implantación: Amplio en entornos corporativos y aplicaciones empresariales; presencia consolidada en el ranking de uso.

PHP
Características: enfocado a web, fácil despliegue, gran cantidad de CMS y frameworks (Laravel, Symfony).
Grado de implantación: Sigue muy extendido en la web tradicional (sitios y CMS); su cuota ha decrecido en tendencias generales pero sigue siendo masivo en producción.

Rust
Características: control de memoria sin garbage collector, seguridad en concurrencia, alto rendimiento. Ideal cuando la seguridad y el rendimiento son críticos.
Grado de implantación: Menor que los principales, pero en ascenso para servicios y componentes de red/infra; gana tracción en sistemas donde el rendimiento y la seguridad importan.

Kotlin
Características: corre en JVM, interoperable con Java, sintaxis moderna; frameworks como Ktor permiten backends ligeros.
Grado de implantación: Creciente en backends JVM y empresas que buscan reemplazar o modernizar bases Java; cada vez más usado en microservicios y APIs.

Ruby (on Rails)
Características: alta productividad para desarrollo rápido.
Grado de implantación: Menor que en su pico histórico, pero todavía usado en startups y productos que priorizan velocidad de desarrollo.

11. Características y posibilidades de desarrollo de una plataforma XAMPP.

XAMPP es un paquete local fácil de instalar que reúne Apache, MySQL, PHP y Perl, pensado para desarrollo, pruebas y prototipado de aplicaciones web en tu máquina; permite ejecutar sitios PHP/BD localmente, depurar, gestionar bases de datos, etc pero no está recomendado para producción sin endurecerlo.

-Características clave:

-Instalación rápida y centralizada: paquete todo-en-uno para Windows/macOS/Linux.
-Orientado a desarrollo local: arranque/paro desde un panel, logs accesibles y configuración sencilla.
-Configurabilidad: puedes editar httpd.conf, php.ini, crear hosts virtuales, activar módulos.
-Portabilidad: se puede instalar en USB/entorno local para demos.
-Interfaz para DB: phpMyAdmin facilita importar/exportar dumps, crear usuarios, índices.
-Soporte para frameworks y CMS: perfecto para WordPress, Laravel, Symfony, Drupal, etc.

-Posibilidades de desarrollo:

-Desarrollar y probar aplicaciones PHP/BD: desde sitios estáticos con PHP hasta aplicaciones completas con autenticación, sesiones, y CRUD.
-Prototipos y pruebas de concepto: montar rápidamente un MVP sin configurar servidores remotos.
-Aprendizaje y enseñanza: ideal para cursos de PHP, SQL y administración web.

-Depuración local: usar Xdebug, logs de Apache/PHP y phpMyAdmin para depurar consultas y código.
-Desarrollo de CMS: instalar y desarrollar sobre WordPress, Joomla o Drupal localmente.
-Testing de integración básica: probar conexiones a DB, envío de mails, rewriting de URLs.
-Uso conjunto con herramientas modernas: puedes integrar Composer para dependencias PHP, usar Git para control de versiones y usar editores como VS Code.
-Generar certificados locales: con OpenSSL crear certificados autofirmados para pruebas HTTPS.

-Limitaciones y precauciones:

-No recomendado para producción: viene con configuraciones por defecto pensadas para facilidad, no para seguridad.
-Seguridad por defecto débil: usuarios, contraseñas y servicios pueden quedar expuestos si se publica. Siempre cerrar puertos y endurecer antes de exponer.
-Escalabilidad limitada: es local y pensado para una sola instancia; no sustituye infraestructura basada en contenedores/cluster en producción.
-Versionado de componentes: actualizar PHP/Apache/MariaDB requiere actualizar XAMPP entero o instalar versiones específicas por separado.
-Rendimiento y recursos: en proyectos grandes puede consumir muchos recursos; para cargas reales es mejor usar ambientes cloud o Docker/Kubernetes.

12. *En qué casos es necesaria la instalación de la máquina virtual Java (JVM) y el software JDK en el entorno de desarrollo y en el entrono de explotación.*

Instala JDK en el entorno de desarrollo (compilar, depurar, construir, herramientas). En producción (entorno de explotación) normalmente basta con instalar sólo la JVM/runtime para ejecutar las aplicaciones, salvo casos concretos donde también se requiere el JDK.

13. *IDE más utilizados (características y grado de implantación actual).*

14. *Servidores HTTP /HTTPS más utilizados (características y grado de implantación actual).*

15. *Apache HTTP vs Apache Tomcat*

16. *Navegadores HTTP /HTTPS más utilizados (características y grado de implantación actual).*

17. *Generadores de documentación HTML (PHPDoc): PHPDocumentor, ApiGen, …*

18. *Repositorios de software – sistemas de control de versiones: GIT , CVS, Subversion, ...*

19. *Propuesta de configuración del entorno de desarrollo para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USED y xxx-WXED.*

20. *Propuesta de configuración del entorno de explotación para la asignatura de Desarrollo web del lado servidor en este curso (incluyendo las versiones): xxx-USEE.*

21. *Realizar un estudio sobre los siguientes conceptos y su relación con el desarrollo de aplicaciones web:*

CMS – Sistema de gestión de contenidos

ERP – Sistema de planificación de los recursos empresariales

22. *Elegir y realizar un estudio y una presentación para la exposición del trabajo sobre una de las siguientes arquitecturas de desarrollo de Aplicaciones Web:*

• MEAN (con MongoDB y con MySQL) • Java EE vs Spring • Microsoft .NET • Angular 7 • Symfony • Laravel • CakePHP • CodeIgniter