MF0492_3
Programació web a l'entorn servidor
(240 h)

UF1844
Desenvolupament d'aplicacions web a l'entorn servidor (90 h)

UF1845
Accés a dades en aplicacions web de l'entorn servidor (90 h)

UF1846
Desenvolupament d'aplicacions web distribuïdes (60 h)

Continguts generals del mòdul

UF1844

Desenvolupament d'aplicacions web a l'entorn servidor

- Introducció a les arquitectures i als llenguatges de programació
- Programació estructurada i models de desenvolupament MVC (Php).
- Introducció a la programació Orientada a Objectes (Php).

Continguts generals del mòdul

UF1845

Accés a dades en aplicacions web de l'entorn servidor

- Disseny BBd relacionals – Model Entitat relació
- DDL o data definition Language (MYSQL - MariaDB)..
- DML o data management Language (MYSQL - MariaDB).
- Introducció a la programació Orientada a Objectes (Php).
- Acces a dades POO (Php).

Continguts generals del mòdul

UF1846

Desenvolupament d'aplicacions web distribuïdes

JSON - XML i Php

Consum de dades de Serveis web I consiltes asíncrones

- API de tercers
- RESTful o Service web de tercers

UF1844

Desenvolupament d'aplicacions web a l'entorn servidor

- Introducció a les arquitectures i als llenguatges de programació

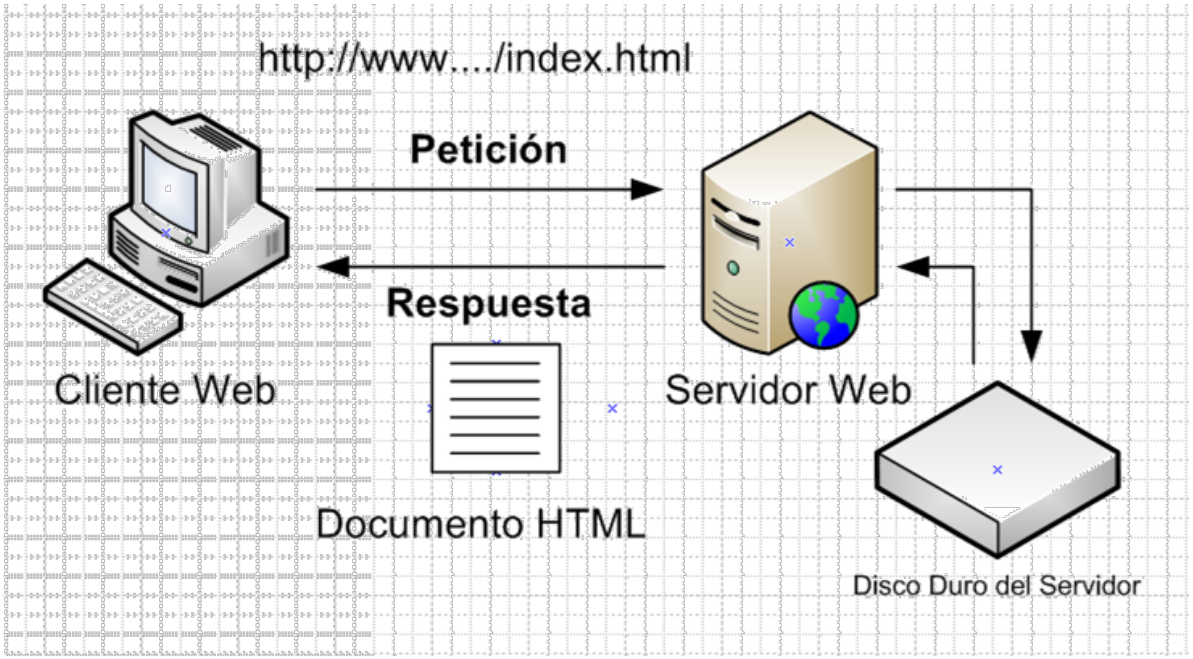
Servidors Web

- Servei de pàgines Web estàtiques
- Servei de pàgines Web dinàmiques

Servidors d'aplicacions

- Llenguatges i entorns comuns
- Arquitectures disponibles
- Model RESTful

Web estàtica 1

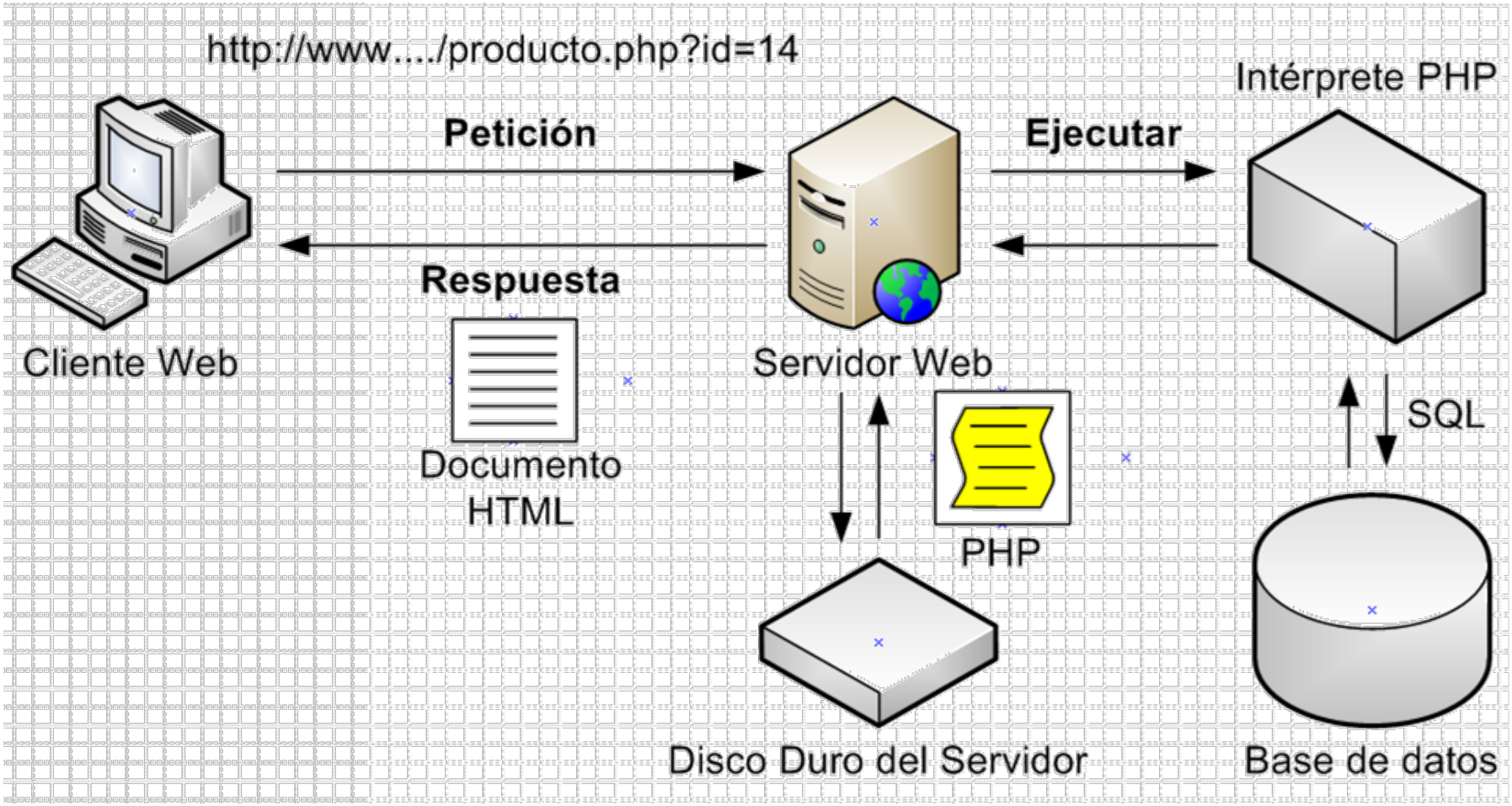


1. Client llança petició web (seguint protocol HTTPS)
2. Servidor llegeix i analitza la petició, descodificant-la
3. Servidor busca arxiu corresponent en disc dur
4. Servidor lliura resposta al client

Web estàtica 2

- Avantatges:
  - Ràpid i de confiança
  - Mateixa petició, mateixa resposta
  - Manteniment uniforme
- Desavantatges:
  - No accepta grans canvis
  - Modificació manual d'enllaços entre pàgines
  - Ampliació costosa: traduccions, índexs, …

Web dinàmica 1



Web dinàmica 2

1. El client llança una petició
2. El servidor la descodifica i descobreix que és una petició dinàmica
3. Cerca el programa a executar i s'ho passa a l'intèrpret o entorn corresponent
4. L'intèrpret o entorn executa el programa, potser interactuant amb altres elements (com una base de dades, peticions a altres equips, …)
5. El servidor obté la pàgina Web de resposta generada i la retorna al client

Web dinàmica 3

- Avantatges:
  - És possible automatitzar tasques complexes i repetitives
  - Es poden generar pàgines a mesura segons les dades subministrades pel client en la petició
  - Es poden emmagatzemar dades en bases de dades, alliberant la necessitat de construir una pàgina Web per producte o concepte
- Desavantatges:
  - Augmenta la complexitat global
  - Més difícil de mantenir

Jekyll, Hugo,…

- Existeix una línia de treball que unifica els dos models en el cas de llocs web de tipus blog
- Consisteix a disposar d'una aplicació instal·lada en un equip que “compila” el lloc web a partir d'arxius senzills (documents *Markdown, fragments HTML, directoris d'imatges).
- Es genera tot un lloc web estàtic que pot ser pujat al servidor mitjançant *FTP.
- Té molts avantatges (manteniment, ampliabilitat, confiabilitat) i alguns inconvenients (no és vàlid en aplicacions dinàmiques directes)

Arquitectures habituals web 1

- Llenguatges interpretats:
  - No requereixen d'un procés de compilació, però si que requereixen d'un programa que els “interpreti”: que els llegeixi, analitzi i executi pas a pas
  - El codi font es “embegut” (inserit) entre línies del codi HTML de les pàgines
  - Són més fàcils de mantenir i són més portables, però més lents en execució comparats amb els compilats
  - Exemples: Perl, PHP, JS, ASP

Arquitectures habituals web 2

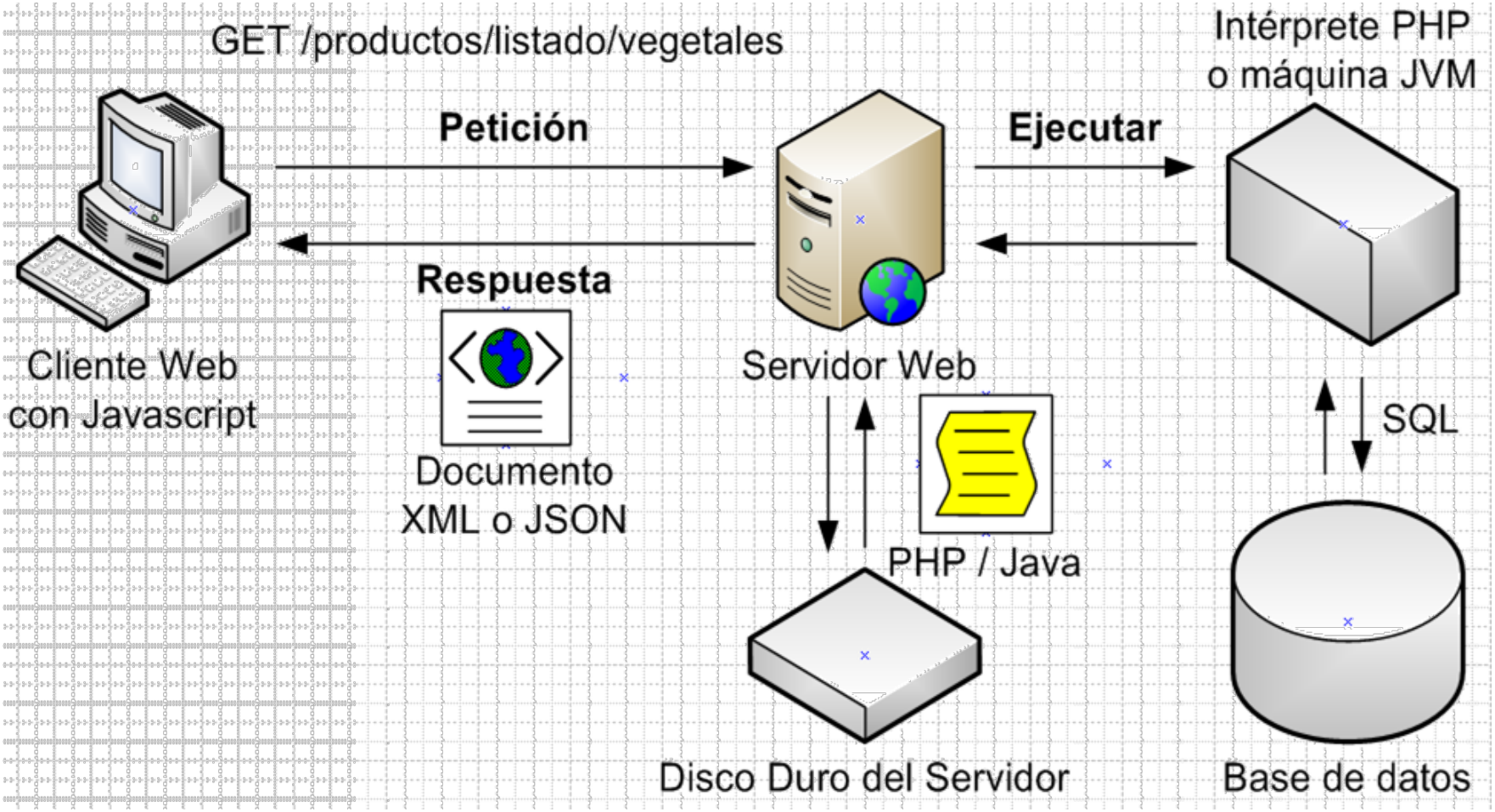
- Llenguatges pseudo-compilats:
  - Es realitza un procés de compilació al que es diu “llenguatge intermedi”. S'executa gràcies a l'existència d'una “màquina virtual” que entén el codi en llenguatge intermedi
  - És més ràpid que els interpretats, però no tant com els compilats
  - És altament portable
  - Requereix d'una infraestructura més complexa en el servidor (màquina virtual, llibreries, …)
  - Exemples: Java, Python

Model RESTful 1

- Existeix una línia actual de treball que millora l'experiència d'usuari i maximitza l'aprofitament del servidor:
  - API (Application Programming Interface)
  - REST (Representational State Transfer)
  - RESTful. API construida amb arquitectura REST

  Es tracta de construir un servei en el costat servidor que respongui a accions senzilles (inserir usuari, llistar productes, modificar usuaris, eliminar usuaris,...) i que retornin al client un arxiu de dades amb els resultats (normalment en XML o JSON).

Model RESTful 2



Model RESTful 3

- Les operacions permeses són 4 mètodes estándard del protocol HTTP: GET, POST, PUT i DELETE. Depenent de la ruta sol·licitada combinen generant una acció concreta:
  - GET /productes/llista
  - DELETE /usuaris/185
- Requereixen que JavaScript (AJAX, Fetch) gestioni l'aplicació en el costat de client invocant les accions necessàries segons convingui i actualitzant dinàmicament la pàgina presentada a l'usuari

Conclusions

- Lloc web petit sense canvis: Web estàtica
- Lloc web tipus blog: generador estàtic (Jekyll)
- Lloc web petit molt canviant (botiga en línia, web corporativa, …): Web dinàmica interpretada (Apatxe + Html + Javascript + PHP + MySQL + API RESTful)
- Lloc web corporatiu / institucional amb gran volum d'informació: Aplicació (Apatxe Tomcat + Html + Javascript + Java + Oracle + Hibernate + API RESTful + model distribuït, …)

Inserció de codi en pàgines Web

- **Requisit previ:** Disposar d'un servidor Web complet amb funcions de PHP i MySQL.
- **Opcions:**
  - Sobre Windows:
    Servidor WAMP (Windows + Apatxe + MySQL + PHP):
    WampServer, o XAMPP
  - Sobre Ubuntu/Linux:
    Afegir Apatxe, PHP i MySQL des del repositori d'Ubuntu.
    sudo apt-get install apache2 php8 mysql-server php8-mysql
  - Sobre Mac US X:
    Habilitar Apatxe i PHP en l'equip i instal·lar MySQL:
    Instruccions detallades