Arquitecturas y tecnologías de programación web

Modelos de ejecución de código en servidores y clientes web

El modelo de desarrollo web se apoya, en una primera aproximación desde un punto de vista centrado en el hardware, en lo que se conoce como arquitectura cliente-servidor 1) que define un patrón de arquitectura donde existen dos actores, cliente y servidor, de forma que el primero es quién se conecta con el segundo para solicitar algún servicio. En el caso que nos ocupa, el desarrollo web, los clientes solicitan que se les sirva una web para visualizarla, aunque también es posible solicitar información si hablamos del caso de los servicios web que también veremos más adelante. En cualquier caso, en ambos casos aparece el mismo escenario, donde un servidor se encuentra ejecutándose ininterrupidamente a la espera de que los diferentes clientes realicen una solicitud.

Normalmente a la solicitud que hacen los clientes al servidor se le llama petición (request) y a lo que el servidor devuelve a dicho cliente le llamamos respuesta (request).

También hay que tener en cuenta que esta arquitectura cliente-servidor plantea la posibilidad de numerosos clientes atendidos por un mismo servidor. Es decir, el servidor será un software multitarea que será capaz de atender peticiones simultáneas de numerosos clientes.

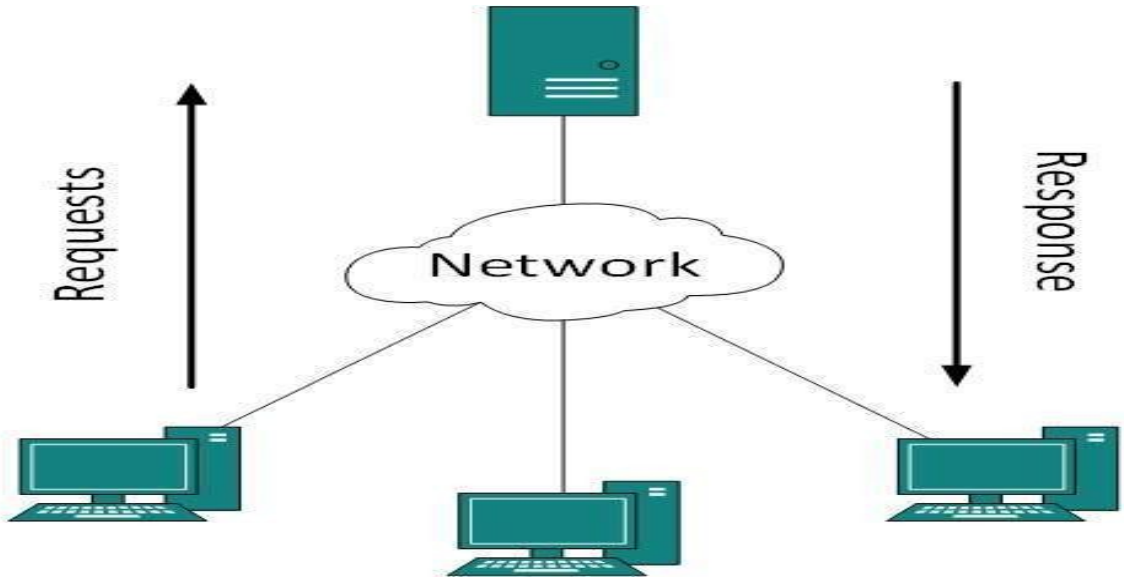

Figure 1: Arquitectura cliente-servidor

Desde un punto de vista de desarrollo una aproximación más detallada para este modelo de ejecución es lo que se conoce como modelo en 3 capas 2). Es un modelo donde se muestra más en detalle como se distribuye el software que participa en cualquier desarrollo web. Sigue estando presente la arquitectura cliente-servidor (todo se basa en ella) pero aparecen más detalles como el software utilizado en cada uno de los dos actores y como interactúan las diferentes tecnologías o aplicaciones.

Para comprender mejor que es el modelo de desarrollo de 3 capas podemos echar un ojo al siguiente esquema donde se muestra cada una de esas capas y de que se encarga cada una de ellas:

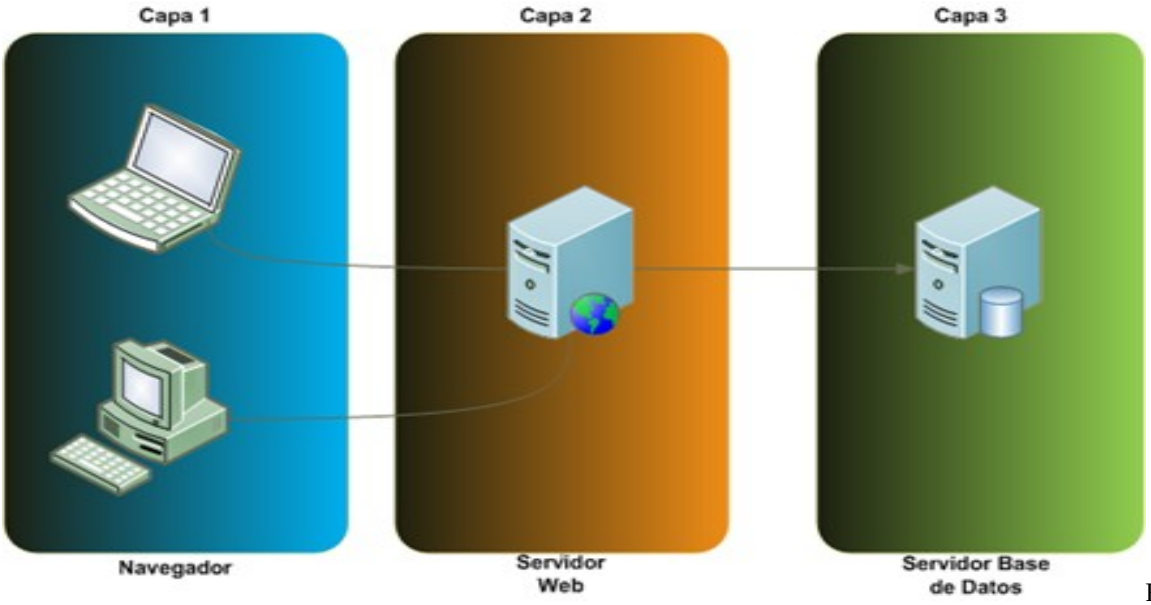

Figure 2: Modelo de desarrollo de 3 capas

- **Capa de presentación**: Es la capa donde la aplicación se muestra al usuario. Básicamente es la (Graphical User Interface, Interfaz Gráfica de Usuario). En el caso de una aplicación web sería el código que se carga directamente en el navegador web. En cualquier caso, se ejecuta directamente en el equipo del cliente.
- **Capa de negocio**: Es la capa intermedia donde se lleva a cabo toda la lógica de la aplicación. Siempre se ejecutará en el lado servidor. Esta capa, tras realizar todos los cálculos y/o operaciones sobre los datos, genera el código que será presentado al usuario en la capa siguiente.
- **Capa de datos**: Es la capa que almacena los datos. Básicamente, en condiciones normales, hace referencia al propio SGBD que es el encargado de almacenar los datos. Dependiendo de la arquitectura de la aplicación, esta capa y la de negocio se pueden encontrar fisicamente en el mismo equipo, aunque también es posible que se tengan que separar por cuestiones de rendimiento. La capa de datos sirve todas la información necesaria a la capa de negocio para llevar a cabo sus operaciones.

  Si nos imaginamos una tienda online, la capa de datos almacena toda la información en una Base de Datos (usuarios, pedidos, articulos, ofertas, . . .), la capa de negocio debe acceder a esa información y, tras procesar un pedido, por ejemplo, debe presentar el resultado final al usuario en el navegador, que es la capa de presentación.

  Y si nos centramos en un caso concreto con software y tecnologías ya definidos, un modelo de 3 capas podría ser el siguiente:

Figure 3: Modelo de desarrollo de 3 capas (desarrollo web con PHP)

- **Navegador web**: En este caso, Mozilla Firefox, Internet Explorer o Google Chrome podrían ser cualquiera de las aplicaciones que ocuparían esta capa
- **Apache + PHP / IIS + ASP**: Un servidor web acompañado de su correspondiente lenguaje de programación web permiten desarrollar la parte que ocupa la capa de negocio. También podríamos colocar la combinación **Apache Tomcat + Servlets**

- **MySQL/PostgreSQL**: Finalmente en la capa de datos podemos colocar cualquier SGBD, como pueden ser *MySQL* o *PostgreSQL*

Front-end, Back-end, Full stack

También teniendo en cuenta en que lado se ubican las tecnologías y para qué se utilizan éstas, actualmente se habla mucho de 3 perfiles diferenciados en el ámbito del desarrollo web:

- **Front-end**: Es la parte del desarrollo que se encarga del diseño y maquetación de la aplicación web utilizando tecnologías como , y Javascript (y sus frameworks). En este caso debe preocuparse también de la correcta presentación en cualquier tipo de dispositivo e incluso del posicionamiento en buscadores
- **Back-end**: Es la parte del desarrollo que se encarga del lado servidor utilizando tecnologías como PHP, JSP y Python. También se encarga de la administración del servidor de aplicaciones y la Base de Datos.
- **Full stack**: En un perfil que engloba los dos anteriores. En este caso el desarrollador quizás no es un experto de ninguna tecnología concreta pero tiene amplios conocimientos de todo el conjunto y es capaz de colaborar en cualquiera de las partes.

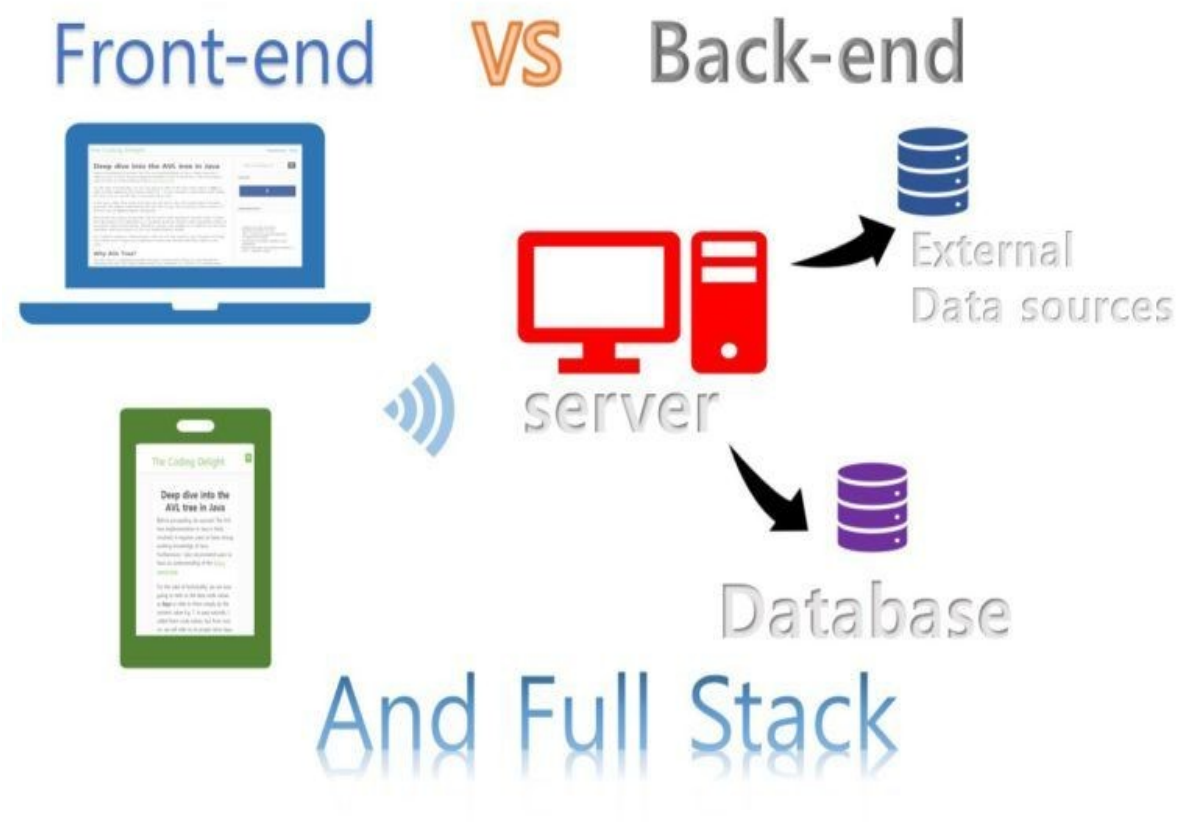

Figure 4: Front-end / Back-end / Full stack

Servidores web y servidores de aplicaciones

Asi como las aplicaciones de escritorio se ejecutan directamente sobre el propio Sistema Operativo, las páginas y aplicaciones web necesitan de una herramienta adicional que permita desplegarlas para su puesta en marcha. Hablamos de servidores web 3) y servidores de aplicaciones 4), respectivamente.

¿Qué es un servidor web?

Un servidor web es una aplicación que recibe una petición HTTP (normalmente a través de un navegador web) y devuelve la página web solicitada (escrita en lenguaje y pudiendo contener código Javascript inscrustado) para que ésta sea interpretada y visualizada por el navegador de quién realizó la solicitud (el usuario).

¿Qué es un servidor de aplicaciones

Un servidor de aplicaciones es una aplicación que contiene una serie de servicios los cuales están accesibles a través de una expuesta a través de Internet. Normalmente los servidores de aplicaciones proporcionan más servicios que los servidores web. Por ejemplo, en el caso de los servidores de aplicaciones para *Java* o *Python*, éstos proporcionan un acceso transparente a la Base de Datos para que el desarrollador se centre exclusivamente en implementar la capa de negocio. Además, pueden proporcionar también servicios como fail-over o balanceo de carga.

Apache

Apache es uno de los servidores web más conocidos. Es software libre y multiplataforma, aunque aproximadamente el 90% de los servidores Apache se ejecutan actualmente en entornos Linux puesto que es el servidor preferido para esta plataforma.

Es muy modular lo que permite incorporar características una vez instalado y puesto en marcha. Eso le hace también muy flexible pudiendo dar servicio a webs escritas en los lenguajes de programación web más extendidos (como PHP, Python, ASP, . . .) a través del módulo correspondiente.

Apache Tomcat

Apache Tomcat es un servidor de aplicaciones que funciona como contenedor de Servlets de Java. Actualmente es capaz de implementar varias especificaciones de Java EE como Servlets y JSP (Java Server Pages) y además proporciona un servidor web *puro* para que se use en combinación con el entorno Java.

WSGI (Web Server Gateway Interface)

WSGI es un interfaz que define como se comunica un servidor web con aplicaciones web o frameworks escritos en Python.

En nuestro caso utilizaremos más adelante el módulo WSGI de Apache para utilizar este servidor web como plataforma para las aplicaciones web que desarrollemos en Python con el framework Django.

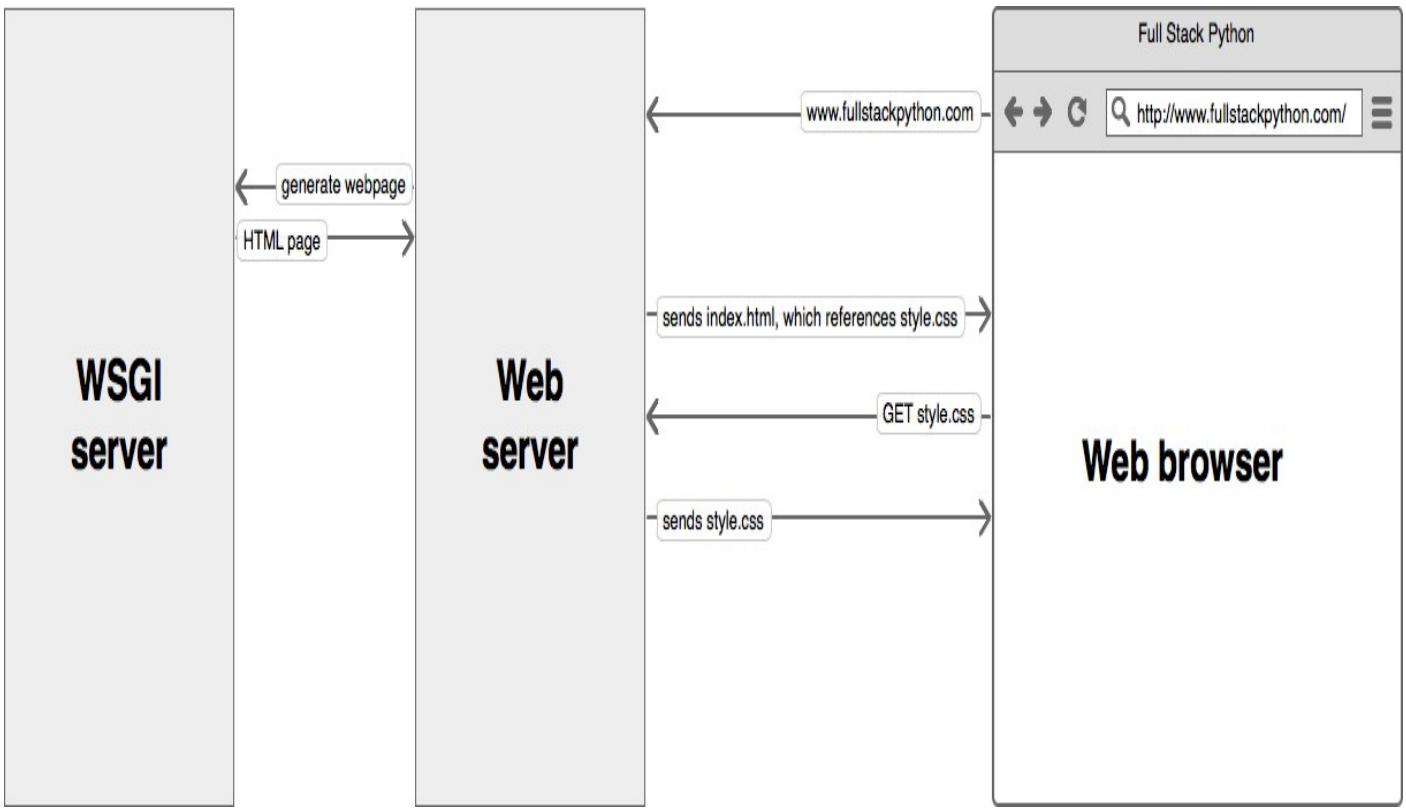

Figure 5: Servidor web Apache e interfaz WSGI

Lenguajes y tecnologías para programación web en entorno servidor

PHP

PHP (PHP Hypertex Preprocessor) es un lenguaje de programación de lado servidor diseñado principalmente para el desarrollo web.

PHP se utiliza como lenguaje de script embebido en páginas y funciona, normalmente, cómo un módulo del servidor web (por ejemplo, en Apache). El servidor web combina los resultados de ejecutar los scripts PHP con el al que va embebido y genera la página web resultante para el navegador.

Actualmente PHP funciona practicamente con cualquier servidor web y en cualquier Sistema Operativo existentes, y casi con cualquier SGBD en caso de que necesitemos utilizar una Base de Datos. A pesar de ello, lo más habitual es verlo formando lo que se conoce como una arquitectura LAMP (Linux, Apache, MySQL y PHP), es decir, funcionando sobre un Sistema Operativo Linux, ejecutándose como un módulo del servidor web Apache y utilizando a MySQL como SGBD para almacenar la información en caso de que se requiera una Base de Datos.

Figure 6: Arquitectura habitual LAMP

El lenguaje *PHP* fue diseñado por *Rasmus Lerdorf* y ahora se mantiene por una comunidad de desarrolladores, y además es open source.

A continuación, un fragmento de una página web dinámica escrita con PHP donde se puede apreciar cómo se incrusta el código junto con el de la página:

```
. . .
. . .
<div id="listado">
  <table class="cebra">
  <thead class="cabecera">
  <tr class="cabecera">
  <td>#</td>
  <td>Nombre</td>
  <td>Descripción</td>
  <td>Stock</td>
  <td>Precio venta</td>
  <td>Imagen</td>
  <td></td>
  <td></td>
  <td></td>
  </tr>
  </thead>
  <tbody class="scroll">
<?php

  $resultado = $bbdd->ejecuta_consulta("SELECT id, nombre, descripcion, stock,
precio_venta, imagen1, miniatura FROM articulos " .
 "WHERE nombre LIKE '%" . $busqueda_rapida . "%' OR descripcion LIKE '%" .
$busqueda_rapida . "%' LIMIT " . $inicio . ", " . $tamano);

  if ($resultado->num_rows == 0) {
    echo "<td colspan='7' style='text-align:center'><span class='titulocelda'>##
Sin Datos ##</span></td>\n";
  }
  else {

    while ($fila = $resultado->fetch_array()) {

      echo "<tr id='registro-" . $fila["id"] . "'>\n";
      echo "<td>" . $fila["id"] . "</td>\n";
      echo "<td>" . $fila["nombre"] . "</td>\n";
```

```
      echo "<td>" . $fila["descripcion"] . "</td>\n";
      if ($fila["stock"] <= 0)
        echo "<td><span class='error'>" . $fila["stock"] . "</span></td>\n";
      else
        echo "<td>" . $fila["stock"] . "</td>\n";
      echo "<td>" . money_format("%i", $fila["precio_venta"]) . "</td>\n";
      echo "<td><a href='articulos/" . $fila["id"] . "/" . $fila["imagen1"] . "'
title='Ampliar'><img src='articulos/" . $fila["id"] . "/" . $fila["miniatura"] .
"'/></a></td>\n";
      echo "<td><a href='run/_generar_barcode.php?id=" . $fila["id"] . "'
title='Generar Código'><img src='icons/barcode16.png' alt='Generar
Código'/></a></td>\n";
      echo "<td><a href='?id=nuevo_articulo&modificar=" . $fila["id"] . "'
title='Modificar Artículo'><img src='icons/editar16.png' alt='Modificar
Artículo'/></a></td>\n";
      echo "<td><a class='eliminar' href='?eliminar=" . $fila["id"] . "'
title='Eliminar Artículo'><img src='icons/cerrar16.png' alt='Eliminar
Artículo'/></a></td>\n";
      echo "</tr>\n";
   }

  $resultado->close();
}
?>
. . .
. . .
```

JSP / Servlets

JavaServer Pages (también conocido como JSP) es una tecnología creada para la creación de páginas web dinámicas del lado servidor. Al igual que PHP, su código se escribe embebido junto con el de la página web y es el servidor de aplicaciones, en este caso, quién debe procesarlo para generar la página web resultante, en .

Por otra parte, Java Servlets es una tecnología que también se puede utilizar para crear contenido web dinámico pero que además extiende su funcionalidad a la posibilidad de conectar esas webs dinámicas con otro contenido accesible a través de Internet. En ocasiones se utiliza junto con JSP para crear aplicaciones web más complejas.

Ambas son tecnologías desarrolladas por *Sun Microsystems* y propiedad ahora de *Oracle*, tras adquirir esta última a la primera hace ya algunos años.

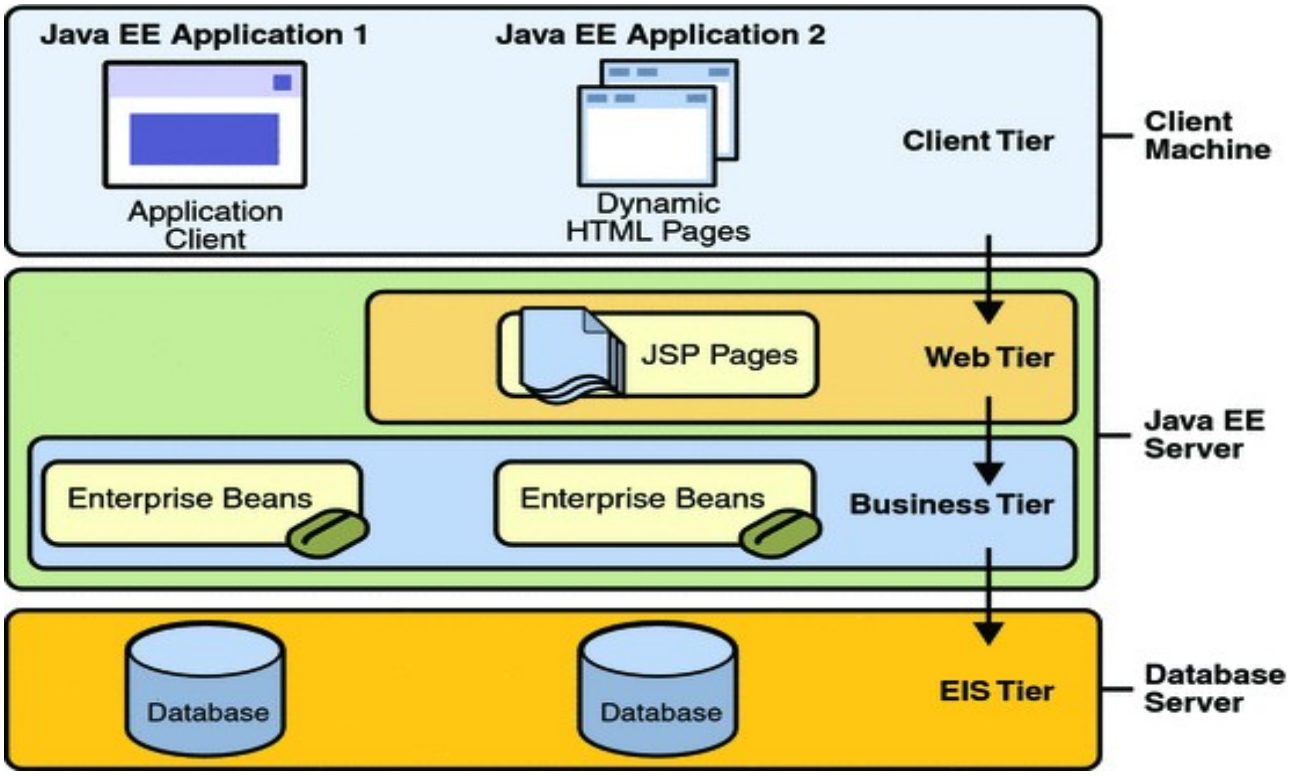

Figure 7: Arquitectura habitual JSP/Servlets

Ejemplo de página web dinámica creada con JSP:

```
<div class="container">
    <div class="page-header">
        <h2>Teléfonos de Emergencia</h2>
    </div>
    <div class="col-md-12">
        <table class="table table-striped">
            <jsp:include page="includes/table_header.jsp">
                <jsp:param name="1" value="Nombre"/>
                <jsp:param name="2" value="Teléfono"/>
            </jsp:include>
            <tbody>
            <%
                Repository repo = new PhoneRepository();
                List<Phone> phonesList = repo.findAll();
                int count = 1;
                for (Phone phone : phonesList) {
            %>
            <jsp:include page="includes/table_body.jsp">
                <jsp:param name="id" value="<%=phone.getName()%>"/>
                <jsp:param name="count" value="<%=count%>"/>
                <jsp:param name="data" value="<%=phone.toString()%>"/>
            </jsp:include>
. . .
. . .
<div class="container" style="margin-left:0px;width:25%">
        <form class="form-anadir" action="PhoneActions">
            <label for="inputName" class="sr-only">Nombre</label>
            <input type="text" name="name" id="inputName" class="form-control"
placeholder="Nombre" required autofocus>
            <label for="inputNumber" class="sr-only">Número</label>
            <input type="text" name="number" id="inputNumber" class="form-
control" placeholder="Número" required>
            <!--<input type="file" id="inputFile" class="form-control"
placeholder="Imagen">-->
            <input type="hidden" name="action" value="insert"/>
```

```
            <button class="btn btn-primary btn-block"
type="submit">Añadir</button>
        </form>
    </div>
. . .
. . .
```

Ejemplo de Java Servlet, correspondiente al mismo proyecto que la página JSP anterior:

```
@WebServlet(name = "PhoneActions")
public class PhoneActions extends HttpServlet {

    private final static Logger LOGGER =
            Logger.getLogger(PhoneActions.class.getCanonicalName());

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        String action = request.getParameter("action");
        Repository repo = new PhoneRepository();

        switch(action) {
            case "insert":
                repo.insert(new Phone(request.getParameter("name"),
                        Phone.IMAGE,
                        request.getParameter("number")));
                response.sendRedirect("phones.jsp");
                break;
            case "delete":
                repo.delete(request.getParameter("name"));
                response.sendRedirect(request.getHeader("referer"));
                break;
            default:
                break;
        }
    }
}
```

Python

5)

El lenguaje *Python* fue diseñado por *Guido van Rossum* y ahora se mantiene gracias a una comunidad de desarrolladores, y es open source.

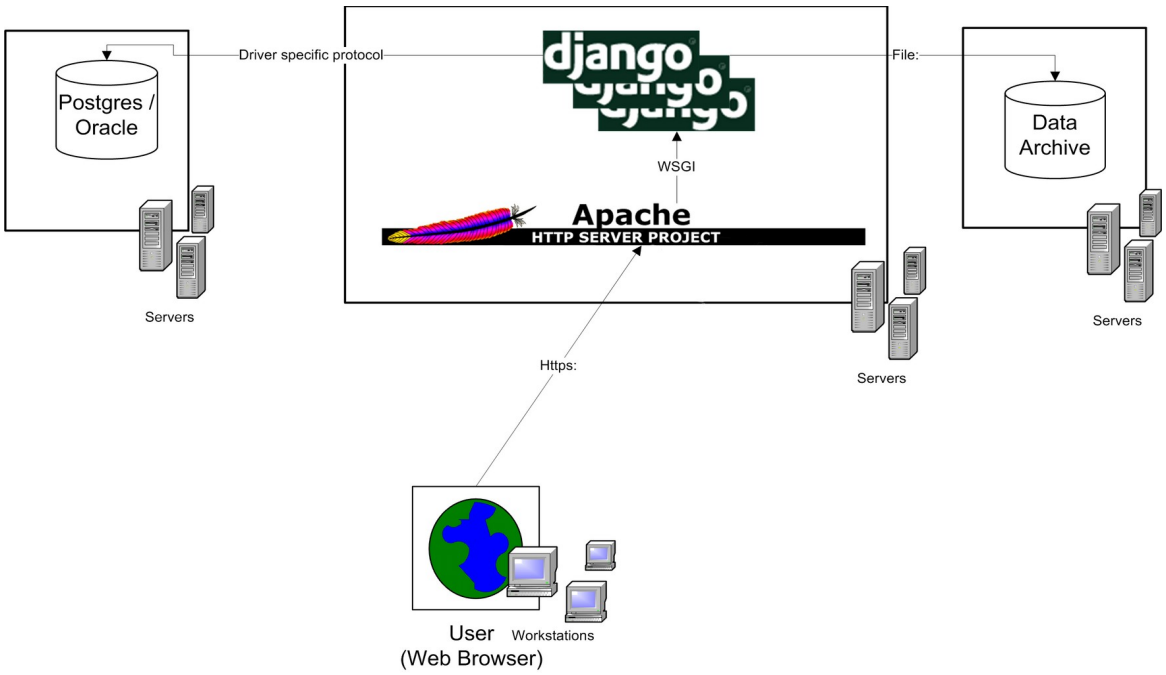

Figure 8: Arquitectura habitual JSP/Servlets

Ejemplo de código escrito con Python. En este caso utilizando el framework *Django* para el desarrollo de aplicaciones web:

En este caso, al tratarse de un framework MVC, tendremos la vista:

```
<h1>Mis películas</h1> <a href="#">+</a>
{% if lista_peliculas %}
   <ul>
       {% for pelicula in lista_peliculas %}
           <li><a href="{% url 'pelicula' pelicula.id
%}">{{ pelicula.titulo }}</a></li>
       {% endfor %}
    </ul>
{% else %}
   <p>No hay películas disponibles</p>
{% endif %}
```

Separada del controlador:

```
from django.shortcuts import render

. . .

def index(request):
    lista_peliculas = Pelicula.objects.all()
    context = {'lista_peliculas': lista_peliculas}
    return render(request, 'mispeliculas/index.html', context)
```

ASP.NET

ASP.NET es una tecnología, creada por Microsoft, para el desarrollo de sitio web dinámicos, aplicaciones y servicios web. Es la tecnología sucesora de lo que antes se conocía como *ASP*, la antigua tecnología de Microsoft para la creación de páginas web dinámicas.

Al funcionar sobre la plataforma *.NET* de Microsoft, permite que se pueda desarrollar en cualquier de los lenguajes de programación de dicha plataforma, *Visual Basic .NET* o *C#*.

Lo más habitual es verlo funcionar junto con el servidor web de Microsoft, IIS (Internet Information Server).

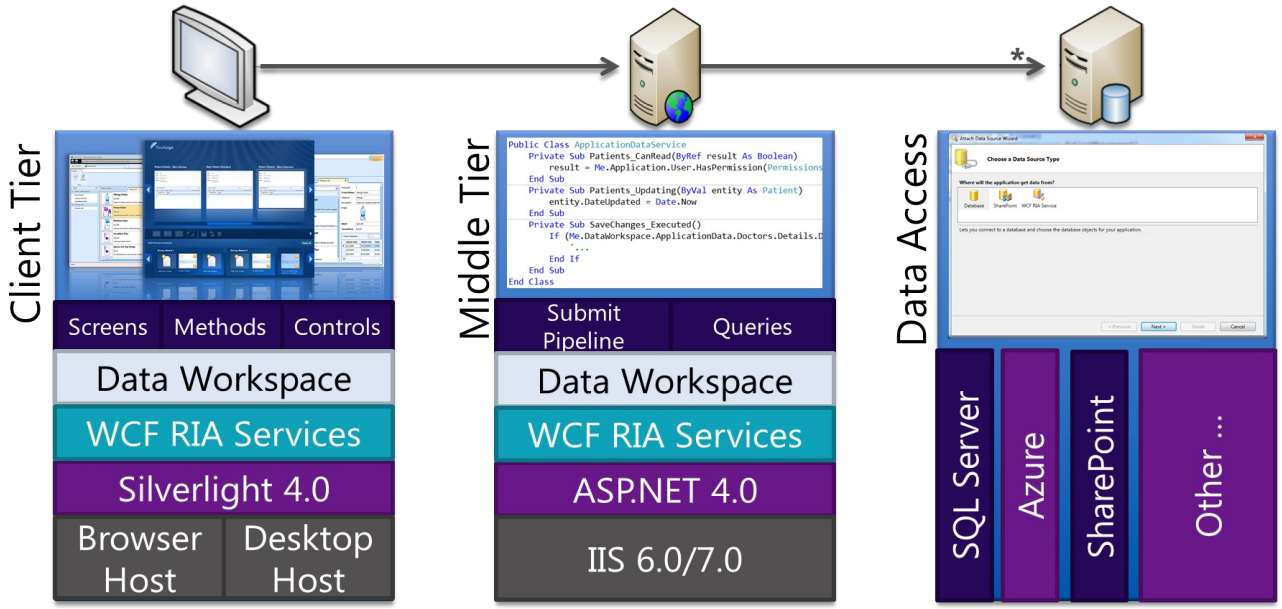

Figure 9: Arquitectura habitual JSP/Servlets

---

1) https://en.wikipedia.org/wiki/Client–server_model

2) https://en.wikipedia.org/wiki/Multitier_architecture

3) https://en.wikipedia.org/wiki/Comparison_of_web_server_software

4) https://en.wikipedia.org/wiki/Comparison_of_application_servers

5) http://www.python.org