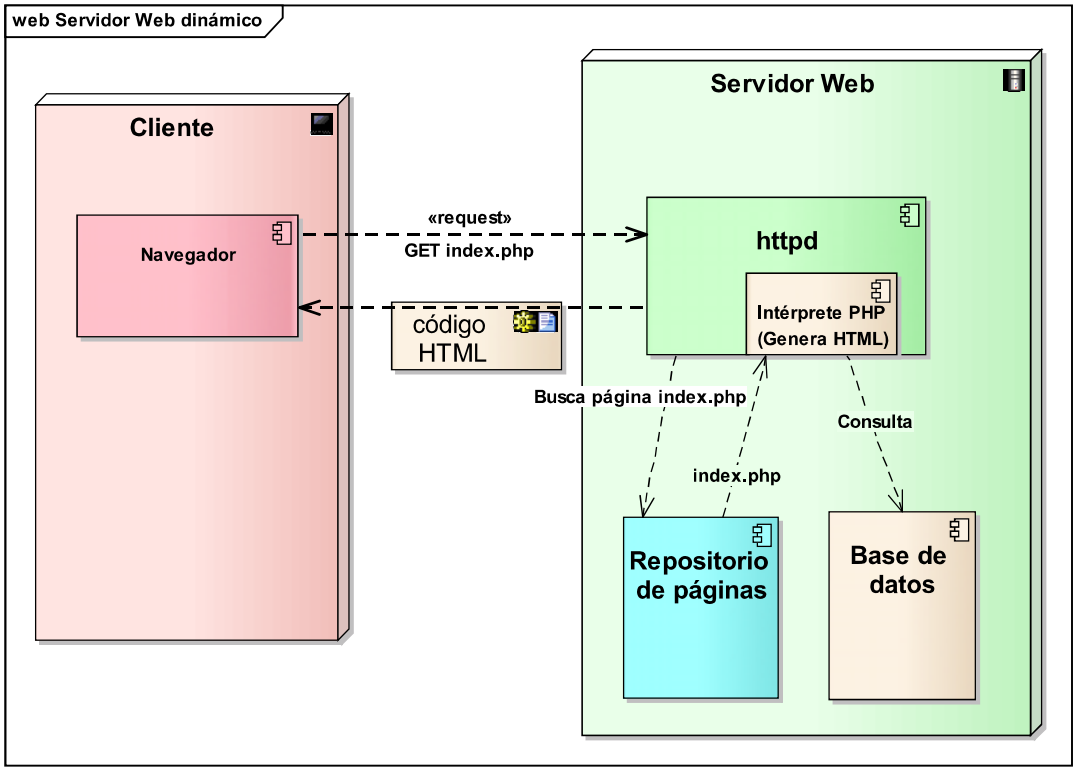
# PHP

## Uso de MySQL con PHP

## Acceso a la base de datos en PHP



## Uso de una base de datos MySQL desde PHP

- Con PHP5 se recomienda utilizar la extensión **MySQLi** (*Mysql improved*) en vez de la tradicional **Mysql**
  - Permite utilizar las mejoras de las últimas versiones del servidor MySQL
  - Interfaz orientada a objetos
- Alternativa: PHP Data Objects (**PDO**)
  - Interfaz ligera para acceso a bases de datos, con soporte para MySQL y otros sistemas de gestión de bases de datos
    - Un driver específico para cada SGBD
  - Proporciona una capa de abstracción para el acceso a datos
    - Independiente del tipo de SGBD
      - No usa la sintaxis SQL
    - Orientado a objetos



## Uso de una base de datos MySQL desde PHP

- Normalmente comprende los siguientes pasos:

1. **Conexión** con el servidor de bases de datos y selección de una base de datos
   - Se obtiene un objeto para operar con la base de datos
2. **Uso de la base de datos**
   - Envío de operación SQL a la base de datos
   - Recepción y tratamiento de los resultados
   - Liberar memoria de resultados
3. **Desconexión**

# Conexión con la base de datos

- Para utilizar una base de datos hay que indicar el servidor y la base de datos que se quiere utilizar, con un usuario

```
$mysqli = new mysqli($hostname, $usuario, $password,$basededatos);
if ( mysqli_connect_errno() ) {
    echo "Error de conexión a la BD: ".mysqli_connect_error();
    exit();
}
```

  - Devuelve un objeto sobre el que operar con la base de datos
  - Si hubiera un error se comprueba con el método `mysqli_connect_errno()`

- Cuando se deja de utilizar la base de datos conviene cerrar la conexión al servidor para liberar recursos ordenadamente

```
$mysqli->close();
```



# Operaciones SQL en una base de datos MySQL

- Las **queries SQL** se pasan con el método **query**

```
$mysqli->query("SQL query");
```

  - Devuelve un objeto que permite tratar los resultados
  - Devuelve FALSE si hay algún error
  - Si se ponen variables PHP en la query, se ponen entre comillas simples para que la función mysql_query las reemplace por su valor

```
$empresa="Empresa%";
$query="SELECT * FROM clientes WHERE nombre LIKE '$empresa'";

$resultado=$mysqli->query($query)
        or die ($mysqli->error. " en la línea ".(__LINE__-1));

$numregistros=$resultado->num_rows;
echo "<p>El número de clientes con nombre Empresa* es: ",$numregistros,".</p>";
```

# Operaciones SQL en una base de datos MySQL

- Varios atributos y métodos de la clase **mysqli_result** facilitan el tratamiento de los registros obtenidos
  - **$num_rows**: Número de registros (filas)

    `$numfilas=$resultado->num_rows;`
  - **fetch_array()** o **fetch_all()**: Devuelve todas las filas en un array asociativo, numérico, o en ambos
    - **fetch_assoc()**: Lo mismo pero como array asociativo

    `$registro=mysqli->fetch_array([modo])`
    - Argumento opcional para indicar cómo se accede a los registros
      - Usando el nombre del campo como índice: MYSQL_ASSOC
      - Usando la posición como índice: MYSQL_NUM
      - Usando tanto el nombre de campo como la posisicón: MYSQL_BOTH
  - **free()**: Libera la memoria asociada al resultado

    `$resultado->free();`



# Operaciones SQL en una base de datos MySQL

- Ejemplo: listado de la tabla clientes

```
$query="SELECT * FROM clientes";

$resultado=$mysqli->query($query)
        or die ($mysqli->error. " en la línea ".(__LINE__-1));

$numregistros=$resultado->num_rows;
echo "<p>El número de clientes con nombre Empresa* es: ",$numregistros,".</p>";

echo "<table border=2><tr><th>NIF</th> <th>Nombre</th> <th>Dirección</th>
<th>Email</th> <th>Teléfono</th></tr>";
while ($registro = $resultado->fetch_assoc()) {
        echo "<tr>";
        foreach ($registro as $campo)
                echo "<td>",$campo, "</td>";
        echo "</tr>";
}
echo "</table>";
$resultado->free();
```

# SQL

- **SELECT**
  - Recupera elementos de una tabla o conjunto de tablas (con JOIN)

```
SELECT campos FROM tabla WHERE campo = valor
```

    - Si se quieren todos los campos, usar *
    - Si se omite la cláusula WHERE se tienen todos los campos de la tabla
    - Para la condición WHERE se pueden usar varios operadores:
      - = <> != < <= > >=
      - AND OR NOT
  - Se pueden recuperar campos de varias tablas

```
SELECT tabla1.campo1 tabla2.campo2 FROM tabla1, tabla2
   WHERE campo3=valor3 AND tabla1.campo1 = tabla2.campo2
```

  - También se pueden usar patrones para las condiciones
    - % indica cualquier subcadena

```
SELECT campos FROM tablas WHERE campo3 LIKE patron
```

    - Ejemplo: `SELECT nombre FROM clientes WHERE nombre LIKE Juan%`
  - Ordenar: ORDER BY
  - Para no tener registros duplicados: DISTINCT

```
SELECT DISTINCT campos FROM tablas WHERE ...
```



# SQL

- **INSERT**
  - Inserta nuevos elementos en una tabla
    - Crea un nuevo cliente

```
INSERT INTO clientes (nif, nombre, direccion, email, telefono)
VALUES ("M3885337J", "Empresa Uno", "Calle Uno, Madrid",
"jefe@empresauno.com", "91 2347898")
```

- **UPDATE**
  - Actualiza campos de una tabla
    - Modifica el importe del producto "Producto1"

```
UPDATE productos SET precio = 399.99 WHERE nombre="Producto1"
```

- **DELETE**
  - Elimina registros de una tabla
    - Elimina pedidos con más de 30 días de antigüedad

```
DELETE FROM pedidos WHERE fecha < CURDATE()-10
```