DESARROLLO WEB EN ENTORNO SERVIDOR

CAPÍTULO 5:

Desarrollo de Aplicaciones Web utilizando código embebido

Marcos López Sanz
Juan Manuel Vara Mesa
Jenifer Verde Marín
Diana Marcela Sánchez Fúquene
Jesús Javier Jiménez Hernández
Valeria de Castro Martínez

Ra-Ma®

Índice

- Control de estado en aplicaciones Web:
  - Cookies.
  - Sesiones.
- ACL`s: Listas de control de acceso .
- Autentificación universal.
  - OpenID.
  - Oauth.
- LDAP: Directorios de Acceso.
  - Validación y acceso a LDAPs desde páginas web.
- Pruebas y Depuración.
  - Tipos de Pruebas.
  - Pruebas Unitarias. PHPUnit.
  - Tendencias en el desarrollo de pruebas.

Control de Estado en Aplicaciones Web

- *App Web: Se navega por varias páginas web.*
- *¿Cómo guardar los datos que ingresa el usuario en cada una de las páginas?*
- *¿Cómo pasar la información para crear la salida final?*

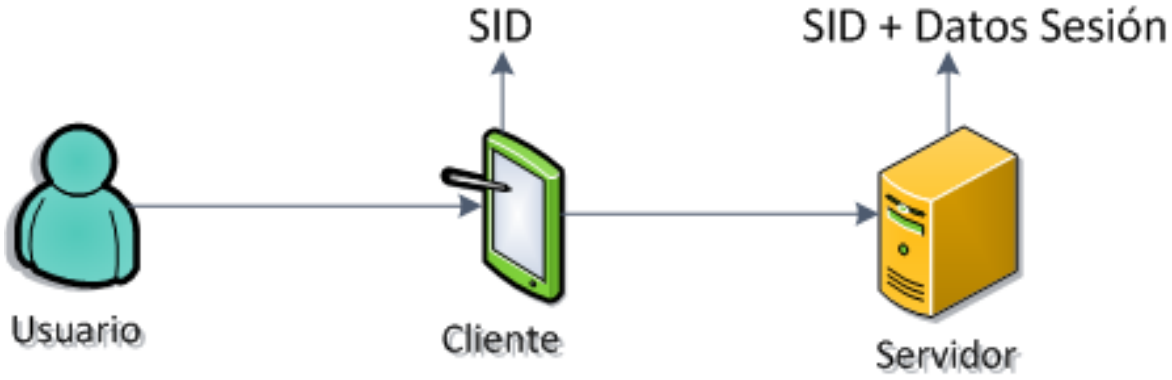
- Objetivo: Garantizar conservación de los datos de un cliente que está realizando una operación en la Web.

Control de Estado en Aplicaciones Web. Sesiones

- Sesión: Mecanismo que guarda información específica de un usuario que está usando una aplicación web.
- Un usuario = una sesión.
- La sesión está vigente durante las páginas web que navega el usuario.
- Cada sesión se identifica con un SID.
- Modalidades:
  - Paso de datos a través de URL: Métodos POST o GET.
  - Cookies.

Control de Estado en Aplicaciones Web. Sesiones

- En el caso de las sesiones normales, el SID es enviado al cliente mientras que en el servidor se guarda toda la info de sesión.
- PHP cuenta con un catálogo de funciones para el manejo de sesiones y de variables de sesión.



Control de Estado en Aplicaciones Web. Sesiones en PHP

- Iniciar una sesión y crear variables de sesión:

```
<?php
//Iniciamos la sesión. Si la sesión ya existe se toma su
   SID
session_start();
echo 'Bienvenido a la página #1';
// El formato para inicializar una variable de sesión es:
// $_SESSION['nombre_var_sesión'] = 'Valor de la
   variable';
// A continuación, iniciamos tres variables de sesión
$_SESSION['favcolor'] = 'verde';
$_SESSION['animal'] = 'gato';
$_SESSION['time'] = time();
?>
```

Control de Estado en Aplicaciones Web. Sesiones en PHP

- `$_SESSION` = Vector con los datos de la sesión.

- Leer valores de variables de sesión:

```
<? php
 $_SESSION["nombre_de_variable_de_sesion "];
?>
```

- Ver si una variable ya existe:

```
<?php
if (isset($_SESION[“nombre_de_variable_de_sesión”])
?>
```

- Destruir una sesión:

```
session_destroy();
```

Control de Estado en Aplicaciones Web. Cookies en PHP

- Para crear una cookie:

```
setcookie(Nombre, Valor, Tiempo_Vida, Path, Dominio,
   Seguro)
```

- Ejemplo:

```
<?php
$value = "algo de algún lugar";
//Una cookie sin parámetros
setcookie("TestCookie");
// Una cookie que expira en 1 hora
setcookie("TestCookie", $value, time()+3600);
?>
```

Control de Estado en Aplicaciones Web. Cookies en PHP

- `$_COOKIE`: Vector con los datos de todas las cookies vigentes dentro de un script PHP.

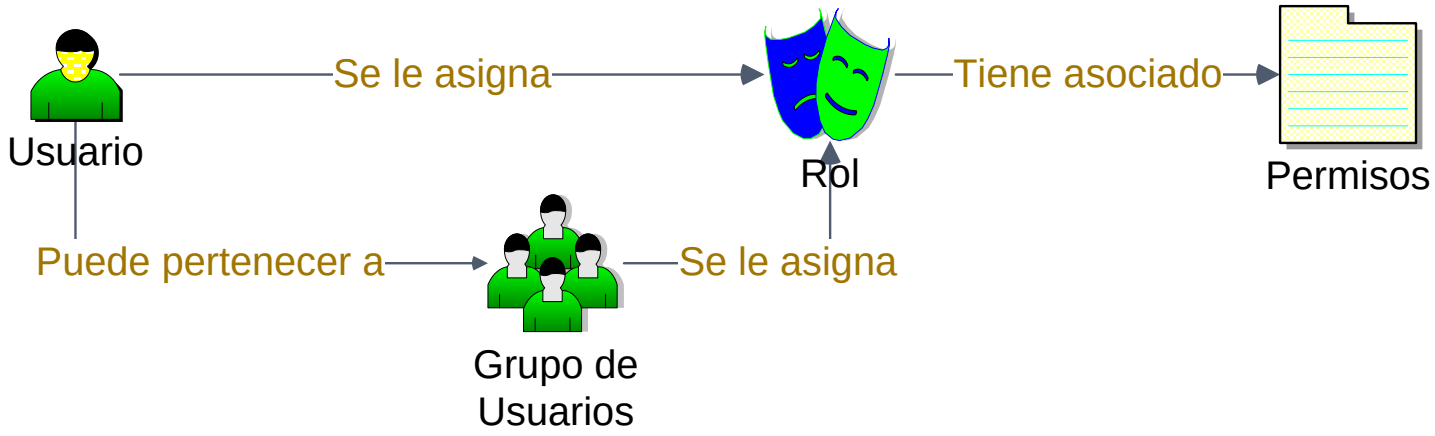
- Leer el valor de una cookie:

```
<?php
    echo $_COOKIE['TestCookies'];
?>
```

- Limitaciones:
  - No debe superar los 4kb.
  - Tener cuidado con datos encriptados.
  - Usar un juego de caracteres estándar. Por ejemplo: ANSI'94.
  - Advertir que la página web usa cookies, para evitar el bloqueo por parte del navegador.

Lista de Control de Acceso
ACL

- ACL: *Access Control List.*
- Busca separar los privilegios de los usuarios.
- Se basa en la creación de (grupos de) usuarios y de roles.

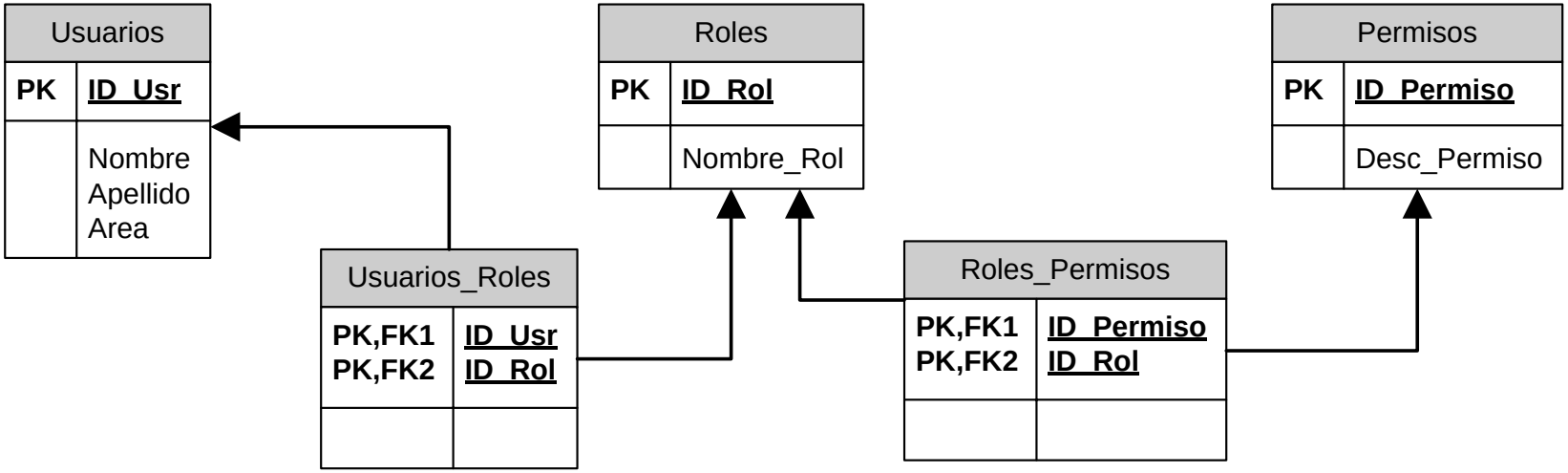


Implementación de un ACL

- Dos pasos importantes:
  - Creación del ACL: Se puede realizar con el apoyo de una base de datos. Se configuran los datos de los usuarios, permisos y roles.
  - Utilización en la página web: Acceso a los privilegios configurados en el ACL a través de funciones web de acuerdo al lenguaje de servidor utilizado.

Implementación de un ACL

- ACL soportado por una base de datos:
  - Una tabla por cada dato relevante: Usuario, Roles, Permisos.
  - Tablas que unen dichos datos a través de sus PK.



Implementación de un ACL

- Conexión entre el ACL y las páginas web:
  - En el caso del que el ACL esté soportado por una BD, se realizarán los mismos pasos de conexión explicados para la conexión entre una página web y una BD (capítulo 6).
  - Una vez realizada la conexión, la página web debe soportar servicios de:
    - Lectura, creación y modificación de usuarios (usando tabla Usuarios).
    - Lectura, creación y modificación de permisos (usando tabla Permisos).
    - Lectura, creación y modificación de roles (usando tabla Roles).
    - Lectura, asignación y modificación de permisos a los roles (usando tabla Permisos_Roles).
    - Lectura, asignación y modificación de roles a usuarios o grupos de usuarios (usando tabla Usuarios_Roles).
    - Estos servicios pueden ser soportados por una clase ACL que maneje todas estas funciones.

Implementación de un ACL

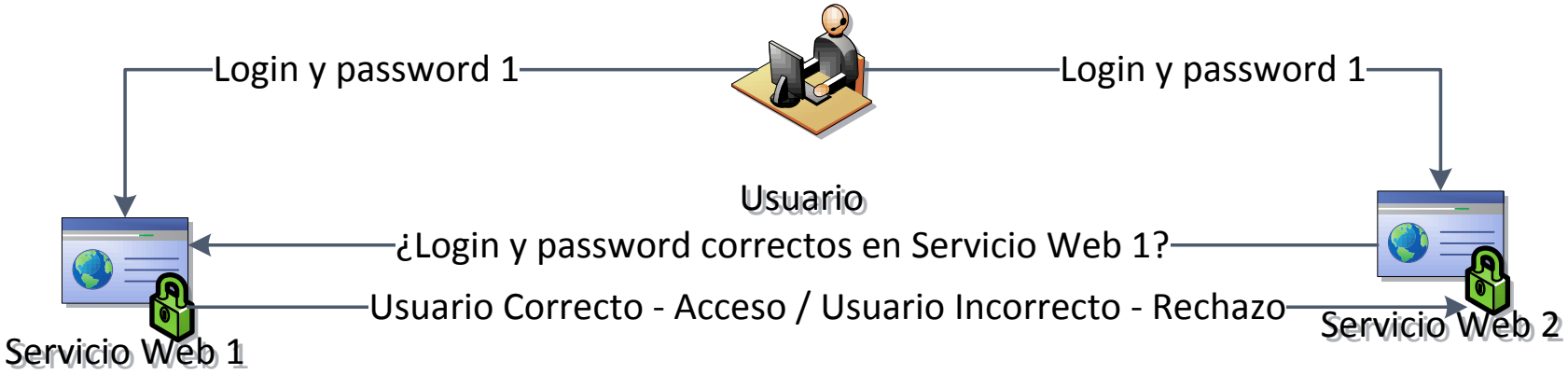
```
<?php
class ACL
{    var $perms = array();//Guarda los permisos de un usuario
     var $userID = 0; //Es el ID del usuario actual
     var $userRoles = array();//Guarda los roles del usuario actual

     function __constructor($userID = '')
     {    if ($userID != '')
          {  $this->userID = floatval($userID);
          } else {
               $this->userID = floatval($_SESSION['userID']);
          }
          $this->userRoles = $this->getUserRoles('ids');
          $this->buildACL();
     }

     function ACL($userID='')
     {    $this->__constructor($userID);  }
} %>
```

Autentificación de Usuarios: OpenID y OAuth

- Avalancha de nombres de usuario y password: el usuario de hotmail, el de yahoo, el de gmail, el de facebook, el de twitter, el de tuenti, etc.
- Usuarios sufren de *fatiga de contraseña.*

Autentificación de Usuarios: OpenID y OAuth

- Solución: Autenticación Universal.
  - Los datos de usuario de una aplicación web sirven para ingresar en otra aplicación web.
- Propuestas:
  - OpenID: http://openid.net
  - OAuth: http://oauth.net



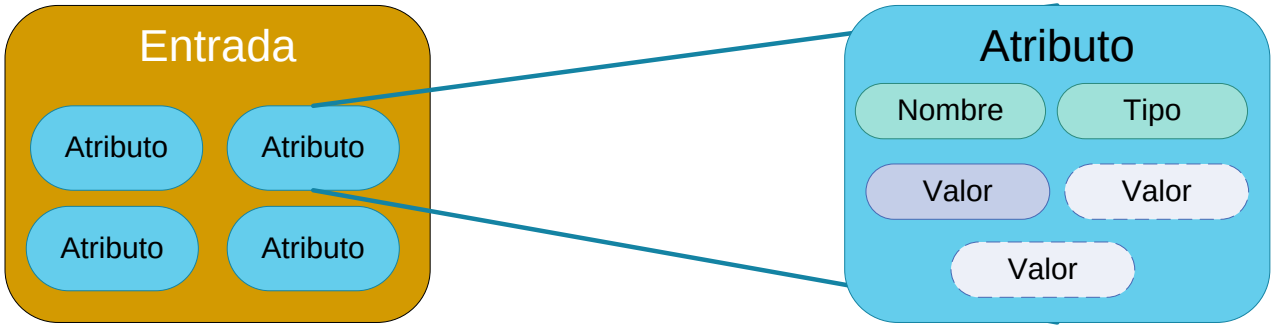
Protocolo de Acceso Ligero al Servicio de Directorios: LDAP

- Directorio: Guía que lista objeto e información sobre los mismos.
- Servicio de directorio: Componentes hardware y software que trabajan para prestar un servicio de búsqueda de información acerca de recursos electrónicos tales como usuarios, impresoras, archivos, etc.
- LDAP: Lightweight Directory Access Protocol.
  - Protocolo a nivel de aplicación para acceso a un directorio ordenado en un entorno de red.
  - Es un sistema cliente/servidor.

Protocolo de Acceso Ligero al Servicio de Directorios: LDAP

- En LDAP un directorio está regido por cuatro modelos:
  - Modelo de información: Estructura.
  - Modelo de nombrado: Nombres e identificación.
  - Modelo funcional: Operaciones posibles.
  - Modelo de seguridad: Protección de la información.
- Modelo de información:
  - Entrada: unidad básica de un directorio.
    - Conjunto de datos acerca de un objeto → Atributos.
    - UID: Identificador único que distingue cada entrada.
  - Las entradas se organizan jerárquicamente a través de un árbol (DIT).

Protocolo de Acceso Ligero al Servicio de Directorios: LDAP

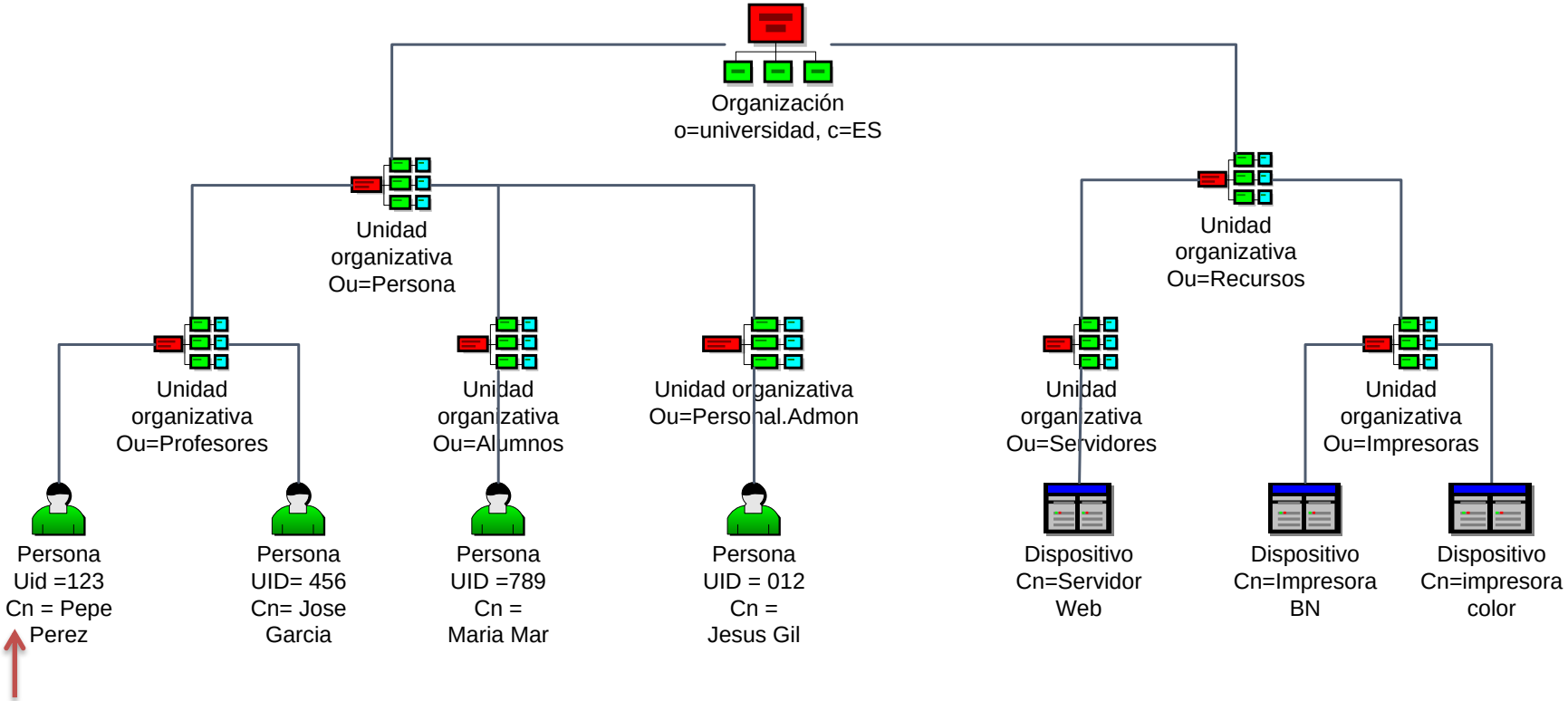


- Cada atributo tiene un nombre, un tipo y al menos un valor.
  - Atributos pueden ser requeridos u opcionales.
- Esquema: Conjunto de atributos permitidos y requeridos. Definen:
  - Tipo de objetos que almacena un directorio.
  - Los atributos de los objetos.
  - El formato de los atributos.
  - Si son opcionales u obligatorios.

Protocolo de Acceso Ligero al Servicio de Directorios: LDAP

- Modelo de nombrado:
  - RDN (Relative Distinguished Name) → Par (atributo=valor). Ejemplo: uid=123.
  - Nombre distintivo (DN – Distinguished name): Nombre único de cada entrada del directorio.
    - Formado por la unión de varios RDN.
    - Se cuenta desde el objeto nombrado hasta la cima del árbol LDAP.
  - DN base: nivel superior de un directorio LDAP.
    - o="nombre_organizacion", c=US. DN base en formato X.500.
    - o=nombre_organizacion.com. DN base derivado de la presencia en Internet de la empresa.
    - dc=nombre_organización, dc=com. DN base derivado de los componentes de dominio DNS de la empresa.

Protocolo de Acceso Ligero al Servicio de Directorios: LDAP



RDN: `uid= 123`

DN: `uid=123456789, ou =Profesores, ou=Persona, o=universidad`

Protocolo de Acceso Ligero al Servicio de Directorios: LDAP

- Modelo funcional: Operaciones para controlar el directorio.
  - ***Bind***: Sirve para autentificarse y especificar una versión del protocolo LDAP.
  - ***Search***: Busca y obtene entradas de directorio.
  - ***Compare***: Prueba si una entrada nombrada contiene un valor de atributo dado.
  - ***Add***: Agrega una nueva entrada al directorio.
  - ***Delete***: Elimina una entrada del directorio.
  - ***Modify***: Modifica una entrada del directorio.
  - ***Modify Distinguished Name*** (DN): Modifica o renombra una entrada.
  - ***Abandon***: Aborta una petición previa.
  - ***Extended Operation***: Es el comando para definir otras operaciones de manera genérica.
  - ***Unbind***: Cierra la conexión.
  - ***Start TLS***: (desde LDAPv3). Activa la extensión Transport Layer Security (TLS) para realizar conexiones seguras.

Validación en un servidor LDAP con PHP

- Activar soporte para LDAP en archivo de configuración `php.ini`. Quitar comentario , es decir, quitar el punto y coma inicial en línea.

```
extension = php_ldap.dll
```

- Reiniciar el servidor de aplicaciones, sea Apache o IIS.
- Ejemplo de secuencia típica en una aplicación que accede un servidor LDAP es la siguiente:
  1. `ldap_connect()`: Establece la conexión al servidor LDAP.
  2. `ldap_bind()`: Autentifica a un usuario, ya sea anonymous o un "login" provisto.
  3. Efectuar una búsqueda o actualización del directorio, desplegar resultados, etc.
  4. `ldap_close()`: Cierra la conexión con servidor LDAP.

Validación en un servidor LDAP con PHP
Búsqueda de una entrada

```
<?php
$ds = ldap_connect("localhost"); // debe ser un servidor LDAP
// Si la conexión es exitosa hacer una autenticación
if ($ds) {
// Registrándonos como anonymous
$r = ldap_bind($ds);
echo "Resultado del bind(): ".$r."<p>";
/* Una vez validados, realizamos un búsqueda en el LDAP. Buscamos los usuarios con uid =
    dsanchez y retornaremos su email */
$sr = ldap_search($ds,"dc=Udec,dc=CL", "uid=dsanchez");
/* Contamos el número de entradas retornado por la búsqueda */
echo "Numero de entradas retornadas: ".ldap_count_entries($ds,$sr)."<p>";
//Obtenemos las entradas retornadas
$info = ldap_get_entries($ds, $sr);
echo "Valor para: ".$info["count"]." itemes retornados:<p>";
for ($i=0; $i<$info["count"]; $i++) {
echo "uid es: ". $info[$i]["uid"] ."<br>";
echo "Entrada email: ". $info[$i]["email"][0] ."<p>";
}
//Cerramos la conexión con el LDAP
ldap_close($ds);
} else {
echo "<h4>No se puede conectar al servidor LDAP</h4>";
} ?>
```

Validación en un servidor LDAP con PHP
Insertar una nueva entrada

```
<?php
$ds=ldap_connect("localhost");
if ($ds) {
/* Nos autentificamos con un usuario con privilegios de modificación
   en el LDAP */
$r=ldap_bind($ds,"cn=root, o=My Company, c=ES", "secret");
//Preparamos la información a insertar en el vector $info.
$info["cn"]="John Jones";
$info["sn"]="Jones";
$info["mail"]="jonj@here.and.now";
$info["objectclass"]="person";
//Agregamos la información al directorio
$r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);
ldap_close($ds);
} else {
echo "No ha sido posible conectarse con el servidor LDAP";
} ?>
```

Pruebas de Software en Aplicaciones Web

- Pruebas:
  - Actividad en la que un componente es ejecutado bajo condiciones y requerimientos específicos.
  - Los resultados se observan y son comparados con los esperados.
  - Garantizan la calidad mínima del software.
  - Valida que el software funcione.
  - Valida que los requisitos hayan sido implementados.
  - Verifica la interacción de los componentes.
- Caso de prueba:
  - Especificación formal de una prueba.
  - Definen el mecanismo de prueba, las entradas y las salidas esperadas.
  - Buen caso de prueba el que detecta un error.

Clasificación de Pruebas

- Pruebas de caja blanca:
  - Permiten examinar la estructura interna del programa.
  - Casos de prueba que garanticen que:
    - Que se ejecutan al menos de una vez todos los caminos independientes de cada módulo.
    - Que se prueben todas las decisiones lógicas en sus ramas verdaderas y falsas.
    - Ejecutar todos los bucles o ciclos con los límites que se les haya definido.
    - Ejecutar las estructuras internas de datos para asegurar su validez.
  - Ejemplo de métodos de caja blanca: Complejidad ciclomática de McCabe.
- Pruebas de caja negra:
  - Complementarias a las de caja blanca.
  - Más frecuentes que las de caja blanca.
  - Se llevan a cabo sobre la interfaz del software.
  - Los casos de prueba de la caja negra buscan demostrar que:
    - Las funciones del software son operativas.
    - Las entradas se aceptan de forma adecuada.
    - Se produce una salida correcta.
    - La integridad de la información externa se mantiene.

Tipos de Pruebas (I)

- De Unidad: Evalúa un módulo (clase, función, etc) concreto.
- Funcionales: Evalúan una funcionalidad completa. Pueden estar implicadas una o varias clases.
- De Integración: Integración de módulos de acuerdo a lo especificado en el diseño.
  - Integración no incremental: Se combinan los módulos por anticipado.
  - Integración incremental: El programa se integra progresivamente y se va probando en pequeños segmentos.
- De Aceptación: Demuestran al cliente que una funcionalidad está correctamente terminada.
- De Seguridad: Verifican que los mecanismos de protección funcionan.

Tipos de Pruebas (II)

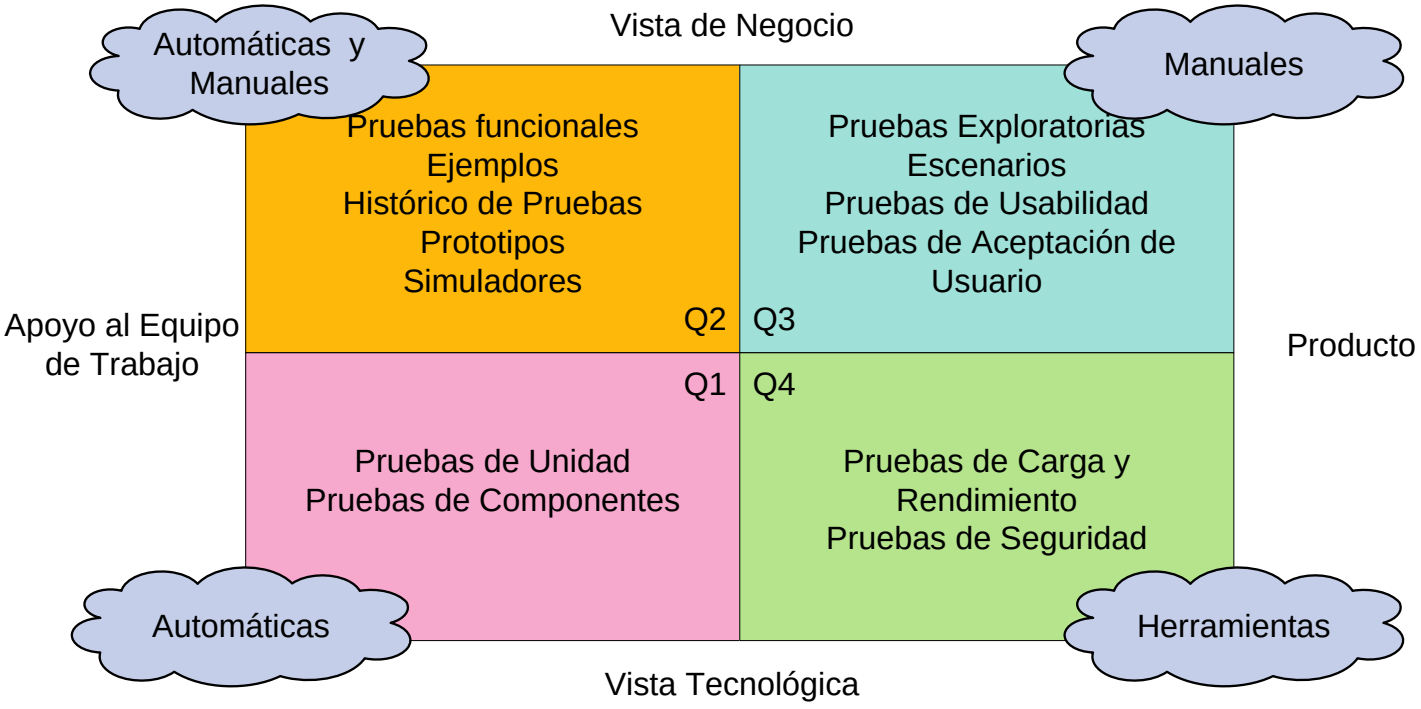
- Del Sistema: Verifican que cada elemento encaja adecuadamente y que se alcanza la funcionalidad y el rendimiento del sistema total.
- De Regresión: Aseguran que los defectos identificados en una prueba anterior se han corregido y que los cambios realizados no han introducido nuevos defectos o reintroducido defectos anteriores.
- De Carga: Determinan el rendimiento del sistema bajo condiciones de carga que se aproximan a la realidad esperada en producción.
- De Volumen: Busca encontrar debilidades en el sistema al momento de manejar grandes volúmenes de datos.

Ejecución de Pruebas

- Pruebas manuales:
  - Sentencias del tipo echo, print, print_r, var_dump para imprimir por pantalla.
  - Silenciar el código con comentarios e irlo activando poco a poco hasta detectar el error.
  - Incluir librerías o rutinas, esperando lograr el resultado esperado.
  - Usar un código encontrado en la red, sin entender lo que se hace.
- Los mecanismos manuales pueden ser útiles si los módulos a probar son pequeños y manejables.
- En entornos grandes y con un volumen de líneas de código muy elevado pueden ser engorrosas y poco prácticas.
- Pruebas automatizadas. Apoyadas en un software.
  - Depurador de código.
  - Analizador de rendimiento.

Ejecución de Pruebas

- Tipos de pruebas y estrategia recomendada:



PHPUnit
Herramienta de pruebas unitarias en PHP

- PHPUnit: Ayuda a probar el código PHP escrito.
  - Disponible en PEAR (http://pear.php.net/).
  - PEAR: Aplicación para distribución de extensiones para PHP.
  - PHPUnit se basa en otras aplicaciones PEAR.
- Casos de Prueba en PHPUnit:
  - Deben ser programados para poder ser reutilizados.
  - ***Recomendación***: El nombre del caso de prueba debe procurar imitar el nombre de la clase o módulo que se pruebe. Ejemplo:
    - Clase a Probar: `conexiónRemota`
    - Caso de Prueba: `TestConexionRemota`
  - Los métodos de prueba deben ser públicos y empezar con la palabra "`test`".
  - Por lo general, una clase de prueba extiende de la clase TestCase de PHPUnit.

PHPUnit
Herramienta de pruebas unitarias en PHP

- Aserción: afirmación que se considera cierta. Útil para la configuración de pruebas.
- Tipos de asersiones:
  - `AssertTrue/AssertFalse`: Comprueba si la entrada es igual a true/false.
  - `AssertEquals`: Comprueba el resultado frente a otra entrada.
  - `AssertGreaterThan`: Comprueba el resultado para ver si es mayor que un valor (también hay `LessThan`, `GreaterThanOrEqual`, y `LessThanOrEqual`).
  - `AssertContains`: Comprueba que la entrada contiene un valor específico.
  - `AssertType`: Comprueba que una variable es de un cierto tipo.
  - `AssertNull`: Comprueba que una variable es nula.
  - `AssertFileExists`: Comprueba que un archivo existe.
  - `AssertRegExp`: Comprueba la entrada con una expresión regular.

PHPUnit
Herramienta de pruebas unitarias en PHP

```
<?php
require_once('RemoteConnect.php');
class RemoteConnectTest extends PHPUnit_Framework_TestCase
{
  public function setUp(){ }
  public function tearDown(){ }
  public function testConnectionIsValid()
  {
    //prueba para asegurarse de que el objeto de un fsockopen es
   válido
    $connObj = new RemoteConnect();
    $serverName = 'www.google.com';
    $this->assertTrue($connObj->connectToServer($serverName)
   !==false);
 }
} ?>
```

PHPUnit
Herramienta de pruebas unitarias en PHP

- Ejecutar pruebas en PHPUnit:
  - Llamar a PHPUnit y señalar el archivo que contiene el código de las mismas. Ejemplo:

```
phpunit /path/to/tests/RemoteConnectTest.php
```

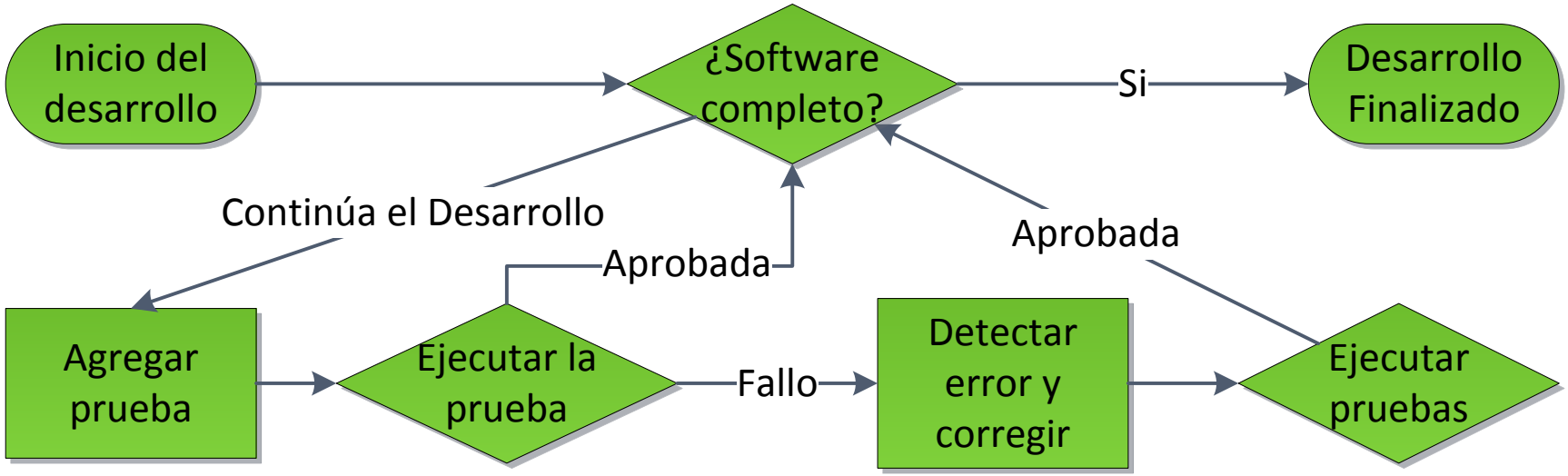
- Salida:

```
PHPUnit 3.4 by Sebastian Bergmann
.
Time: 1 second
Tests: 1, Assertions: 1, Failures 0
```

- Si éxito: Puntos. Número de puntos = numero de pruebas.
- “F” : error.
- “I”: incompleta.
- “S”: omitida (Skipped).

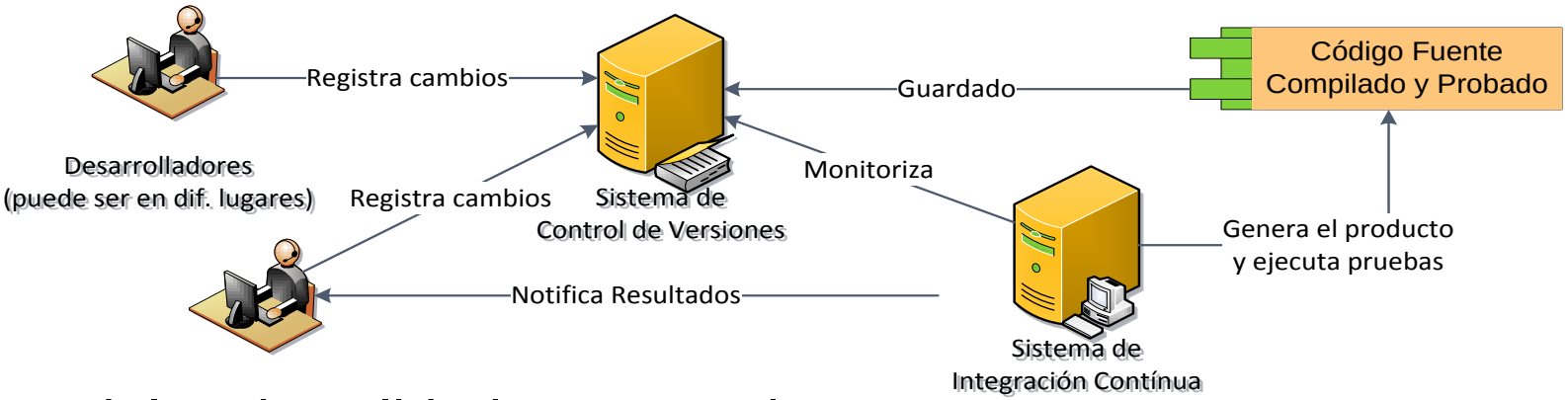
Tendencias en el desarrollo de pruebas
Desarrollo basado en pruebas

- TDD: Test Driven Development.
  - Escribir las pruebas antes que el código.
  - Las pruebas deben comprobar una funcionalidad prevista.
  - Basado en la realización de pruebas unitarias.



Tendencias en el desarrollo de pruebas

- Integración continua: integraciones automáticas frecuentes para detectar errores.



- Modelos de calidad para pruebas:
  - Inspirados en los modelos de madurez como CMMI.
  - TMM: Modelo de Madurez de Pruebas.
  - TMMi: TMM integrado. EEUU. Ofrece 5 niveles de madurez.
  - TPI: Proceso de mejoramiento de pruebas. Europeo. Cercano a niveles de capacidad.