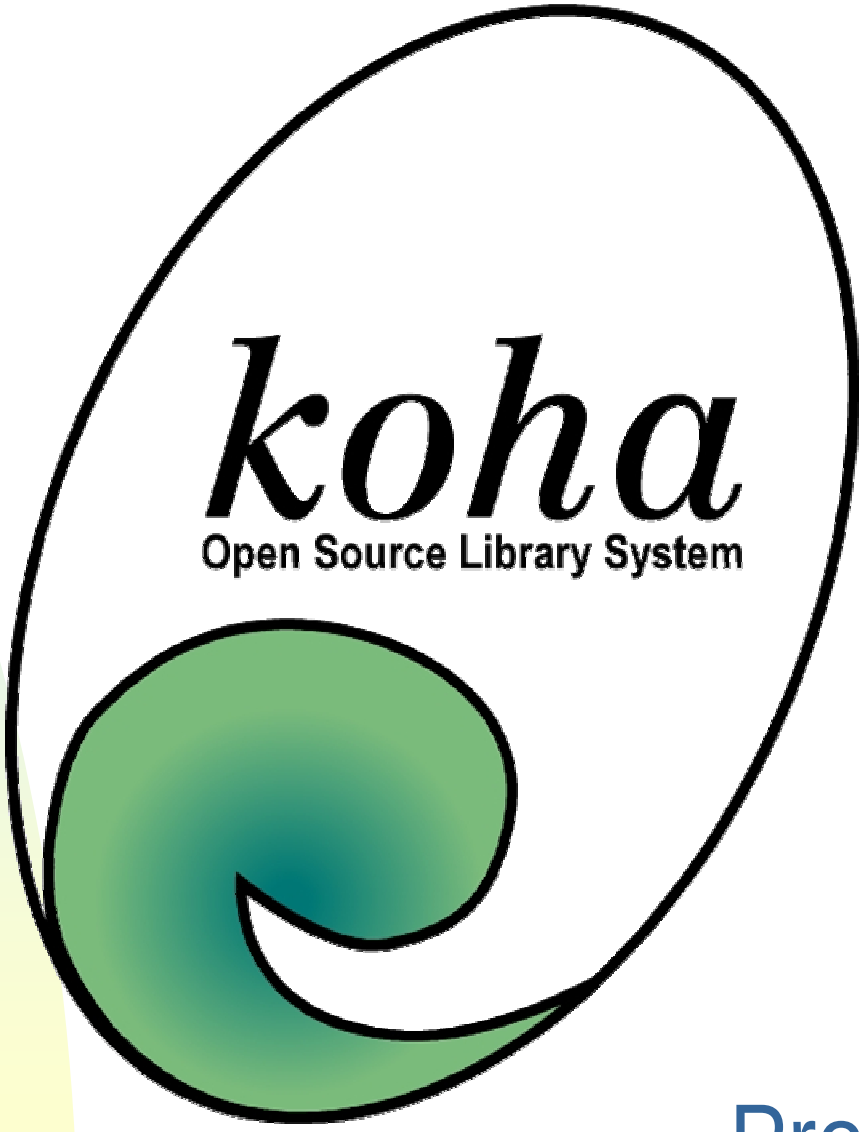


Proyecto KOHA

Grupo de desarrollo UNLP

Integrantes

Equipo interdisciplinario:

- SIU y CESPI: Javier Díaz y Emiliano Marmonti
- Biblioteca Pública: Norma Mangiaterra y Lorena Miranda
- LINTI, Facultad de Informática: Einar Lanfranco, Matías Pagano, Luciano Iglesias y Nahuel Lofeudo
- Facultad de Ciencias Económicas: María Fernanda Pietroboni

Resumen del día

- **Koha**
  - Introducción a Koha
  - Instalación en Linux
- **Herramientas de desarrollo:**
  - Mantis
  - Subversion
- **Perl**
  - Introducción a la programación
  - Koha: Estructura y Ejemplos
- **Modificaciones**
  - Adición de Funcionalidades
  - Solución de Bugs

Proyecto KOHA

Grupo de desarrollo UNLP

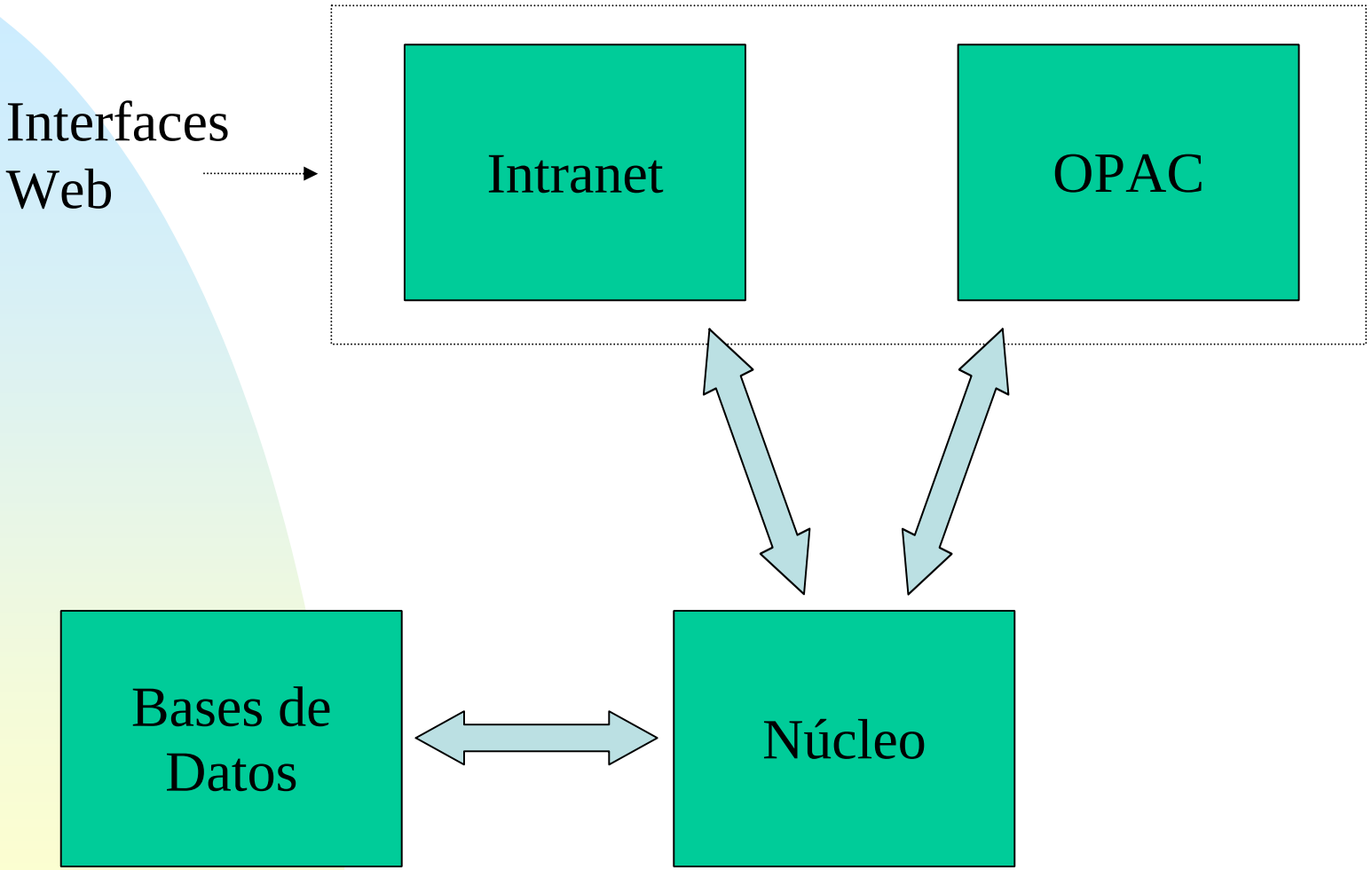
Introducción a Koha

- ***Sistema de Gestión integral de Bibliotecas***
- ***Formado por:***
  - Interfaz Web para Administración
  - Interfaz Web para Acceso Público
  - Bases de datos
  - Núcleo del Sistema

Proyecto KOHA

Grupo de desarrollo UNLP

Funcionamiento de KOHA



Proyecto KOHA

Grupo de desarrollo UNLP

Instalación

- ***Requerimientos:***
  - ***Sistema Operativo***
  - ***Servidor Web***
  - ***Base de Datos***
  - ***Interprete Perl***
  - ***Módulos Perl***
  - ***Koha***
- En este caso utilizaremos Sistema Operativo GNU/Linux Red-Hat, Servidor Web Apache y Motor de Base de Datos MySQL.

Perl

- ***Utilizaremos la versión 5.0.8 de Perl.***
- ***Módulos requeridos:***
  - Date::Manip
  - DBD::mysql
  - DBI
  - Gettext
  - HTML::Template
  - Mail::Sendmail
  - Marc::Record
  - Net::Z3950
  - Set::Scalar
  - Time::Hires

Instalación: Paquetes necesarios

- ***Posicionarse sobre /nethome/koha***
  - cd /nethome/koha
- Instalar y levantar el Apache
  - rpm -ivh httpd-2.0.40-8.i386.rpm
  - service httpd start
- Instalar y levantar el Mysql
  - (ya instalado) rpm -ivh mysql-3.23.52-3.i386.rpm
  - rpm -ivh mysql-server-3.23.52-3.i386.rpm
  - rpm -ivh mysql-devel-3.23.52-3.i386.rpm
  - service mysqld start

Instalación: Paquetes (cont.)

- ***Instalar el Perl***
  - ***(ya instalado) rpm -ivh perl-5.8.0-55.i386.rpm***
- Dependencias de los módulos
  - rpm -ivh ncftp-3.1.3-6.i386.rpm
  - rpm -ivh elinks-0.3.2-1.i386.rpm
  - rpm -ivh zlib-devel-1.1.4-4.i386.rpm
  - rpm -Uvh tcp_wrappers-7.6-34.i386.rpm
  - rpm -ivh libyaz-2.0.15-1.i386.rpm
  - rpm -ivh libxml2-devel-2.4.23-1.i386.rpm
  - rpm -ivh libyaz-devel-2.0.15-1.i386.rpm

Instalación: Módulos (cont)

- ***Módulos Perl***
  - cd ../modulos
- ***Uno a uno por cada módulo:***
  - cd nombre_de_módulo
  - perl Makefile.PL
  - make
  - make test
  - ***make install***
  - ***cd ..***

Atajo: funciona en la misma línea:

perl Makefile.PL; make; make test; make install

Instalación: KOHA (cont)

- ***KOHA***
  - cd ../koha-2.0.0
  - ./installer.pl
    - Este script automaticámente:
      - » Copia los archivos de Koha
      - » Crea la Base de Datos
      - » Crea el archivo de Configuración
      - » Crea el adicional para el servidor web

Instalación: KOHA (cont)

- ***KOHA***
  - cd ../koha-2.0.0
  - ./installer.pl
    - Este script automaticámente:
      - » Copia los archivos de Koha
      - » Crea la Base de Datos
      - » Crea el archivo de Configuración
      - » Crea el adicional para el servidor web

Configuración

- ***Configurar Koha***
  - ***Editar***
    - ***/etc/koha-httpd.conf***
    - ***y sacar el comentario (#) a la línea***

      #Listen 8080

      **** En el caso que hayan elegido los puertos por defecto***
- ***Configurar y reiniciar Apache***
    - ***en el archivo /etc/httpd/conf/httpd.conf agregar la línea Include /etc/koha-httpd.conf***
    - ***service httpd restart***

Herramientas de desarrollo

- Reporte de bugs: Mantis
  - Reporte de fallos
  - Características de los reportes
- Control de versiones: Subversion
  - Clientes de línea de comandos
  - Clientes gráficos

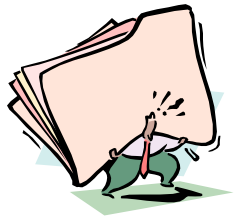
Desarrollo concurrente

- Problemas:
- Comunicación
  - Usuario -> desarrollador
  - Desarrollador -> desarrollador
- Coordinación
  - Asignación de tareas

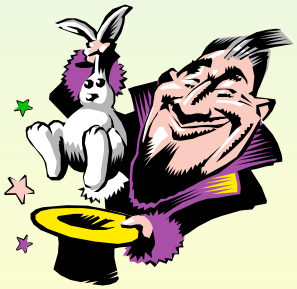
Ciclo de vida de un error

Reporte

Confirmación



Asignación



Resolución

Proyecto KOHA

Grupo de desarrollo UNLP

Reporte de errores

- Usuario
- Proyecto
- Descripción
- Reproducibilidad
  - Pasos para reproducir el error
- Prioridad
- Severidad
  - Funcionalidad <-> Bloqueo
- Comentarios

Reproducible

- Siempre
- A veces
- Aleatorio
- No se ha intentado
- No es posible duplicar
- No se sabe

Gravedad

- Funcionalidad
- Personalización
- Trivial
- Texto
- Ajuste
- Menor
- Mayor
- Cuelgue
- Bloqueo

Prioridad

- Ninguna
- Baja
- Normal
- Alta
- Urgente
- Inmediata

Estado de un reporte

- Nuevo
- Se necesitan más datos
- Aceptado
- Confirmado
- Asignado
- Resuelto
- Cerrado

Resolución de un error

- Abierto
- Arreglado
- Reabierto
- No se pudo duplicar
- No es corregible
- Duplicado
- No es bug
- Suspendido
- No se arreglará

Confirmación

- Un desarrollador comprueba que el error exista
- Si se pudo reproducir se asigna
- Si no se puede reproducir
  - Se deja marcado
  - NUNCA se borran reportes

Facilidades

- Notificación por e-mail
  - Al asignarse un error
    - Desarrollador
  - Al cambiar el estado de un reporte
    - Desarrollador
    - Informador
- Monitorizar error
- Re-abrir error

Ejemplo de reporte

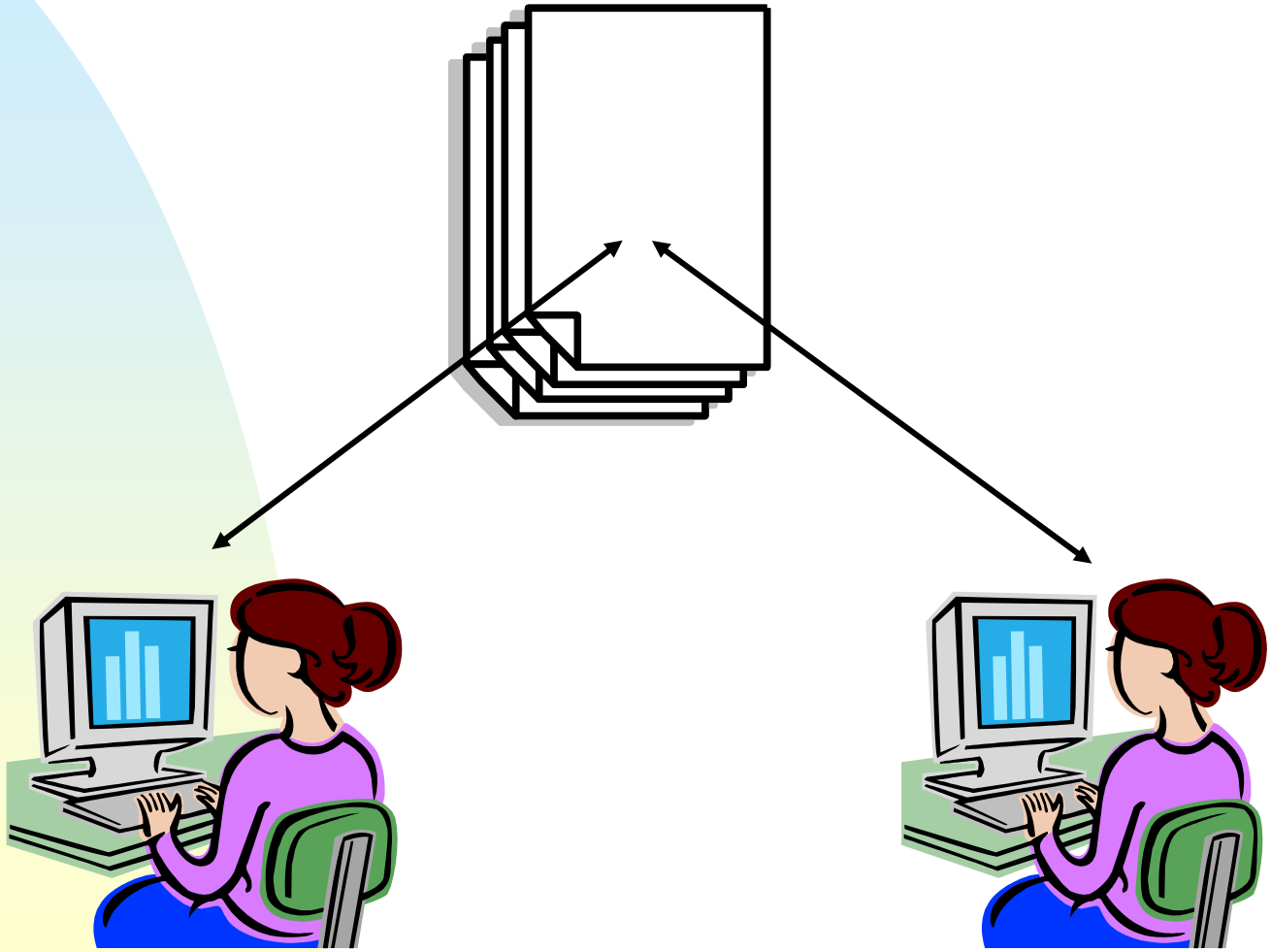
Proyecto KOHA

Grupo de desarrollo UNLP

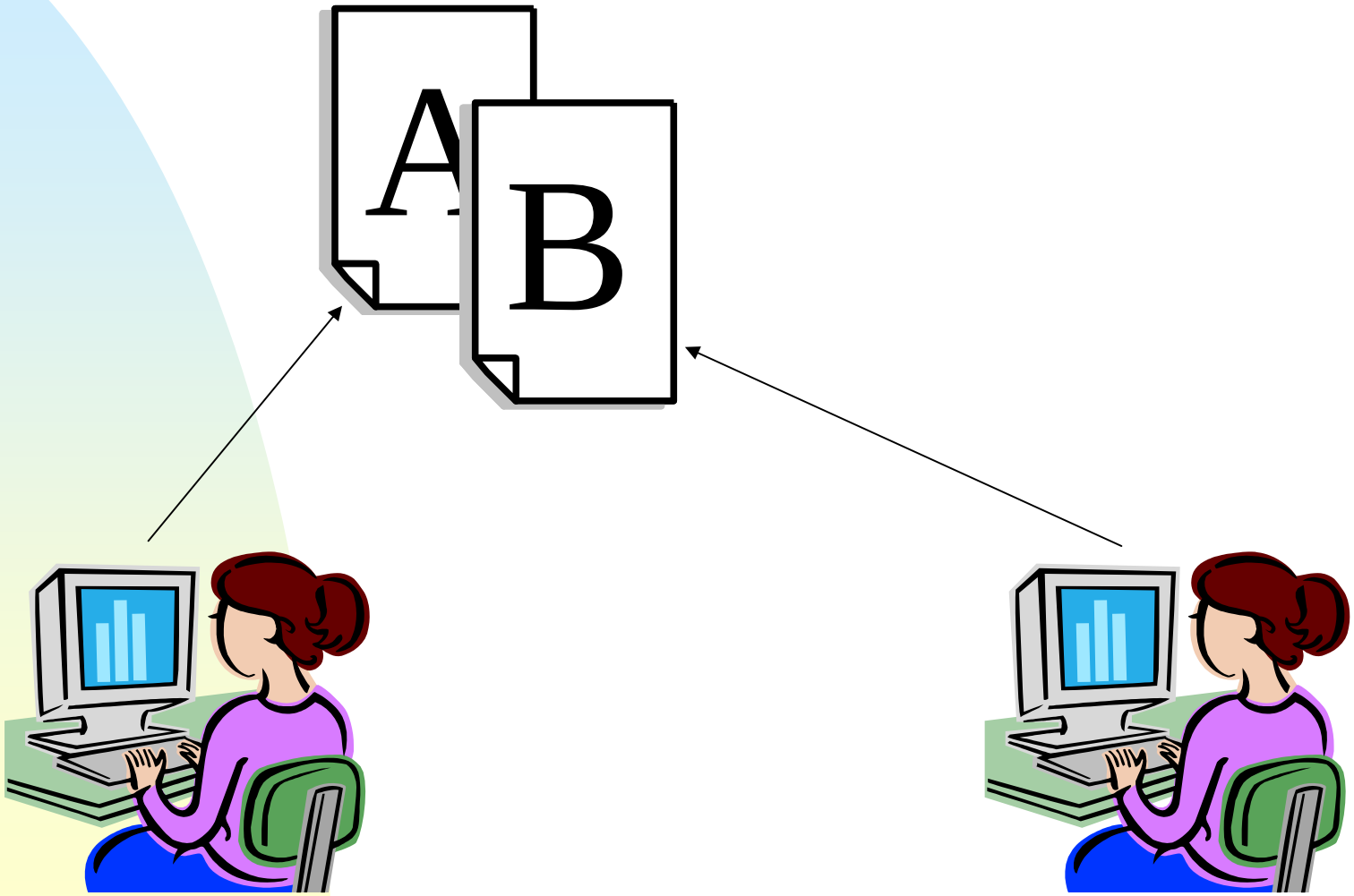
Desarrollo colaborativo

- Potencialmente miles de archivos
- Múltiples desarrolladores
- Físicamente distribuídos
- Poca comunicación

Única copia global



Al hacer modificaciones...



Solución: Subversion

- Repositorio central
  - Única versión “oficial”
  - Los cambios forman un historial
- Una copia para cada desarrollador
- Los cambios de los desarrolladores pueden mezclarse

Características de SVN

- Repositorio versionado
- Transacciones atómicas
- Múltiples protocolos de acceso
- Múltiples tipos de clientes
  - Linux / Unix
    - Consola
    - GUI
  - Windows
- Fácil de usar
- Seguro

Operaciones

- Check-Out (co)
- Commit (ci)
- Actualización (update)
- Operaciones de archivos/directorios
  - ABM de archivos/directorios
  - Meta-información
- Información administrativa (info)
- Historial de operaciones (log)

Ciclo de trabajo: Check-out

```
# export LANG=en_US.UTF-8 (sólo si es necesario)
# svn co http://bugs.linti.unlp.edu.ar/svn/repos koha
A  koha/log
A  koha/opac
A  koha/opac/htdocs
A  koha/opac/htdocs/join.html
A  koha/opac/htdocs/index.html
A  koha/opac/htdocs/opac-tmpl
A  koha/opac/htdocs/opac-tmpl/default
[...]
A  koha/intranet/cgi-bin/detail.pl
A  koha/intranet/cgi-bin/jmemberentry.pl
Checked out revision 1.
#
```

El directorio .svn

- Guarda información administrativa
- Sirve para saber qué se cambió
- Existe uno en cada directorio

NO SE DEBE TOCAR

Commit

- Transmite los cambios al repositorio
- Crea una nueva revisión
- Sólo para usuarios autorizados
- Sintaxis:

#svn ci [-m “mensaje”]

Update

- Sincroniza la copia local con el rep.
- El más simple de todos:
- # svn update
- Acciones:
  - U -> Updated
  - A -> Added
  - D -> Deleted
  - R -> Replaced
  - G -> merGed
  - C -> Conflict

Conflictos

```
main() {
   printf (“hola!”);
 }
```

Original

```
main() {
   printf (“hola!”);
   printf (“mundo”);
 }
```

Pablo

```
main() {
   printf (“hola!”);
   printf (“planeta”);
 }
```

Andrea

Al hacer el checkin

- Pablo:

```
# svn ci
U  main.c
Updated to revision 2
```

- Andrea:

```
# svn ci
C main.c
Updated to revision 3
```

Resolución

- Archivo main.c:

```
main() {
    printf (“hola!”);
    <<<<<<<<<<<< .mine
    printf (“mundo”)
    ============
    printf (“planeta”);
    >>>>>>>>>>>> .r2
 }
```

- Copia local (main.c.mine)
- Copia de la versión anterior (main.c.r1)
- Copia de la versión actual en SVN

Resolución (cont.)

- Revertir los cambios:
  - # svn revert
- Sobreescribir el main.c
- Resolver el conflicto manualmente
- En cualquier caso:
  - #svn resolved main.c

Comandos misceláneos

- # svn log

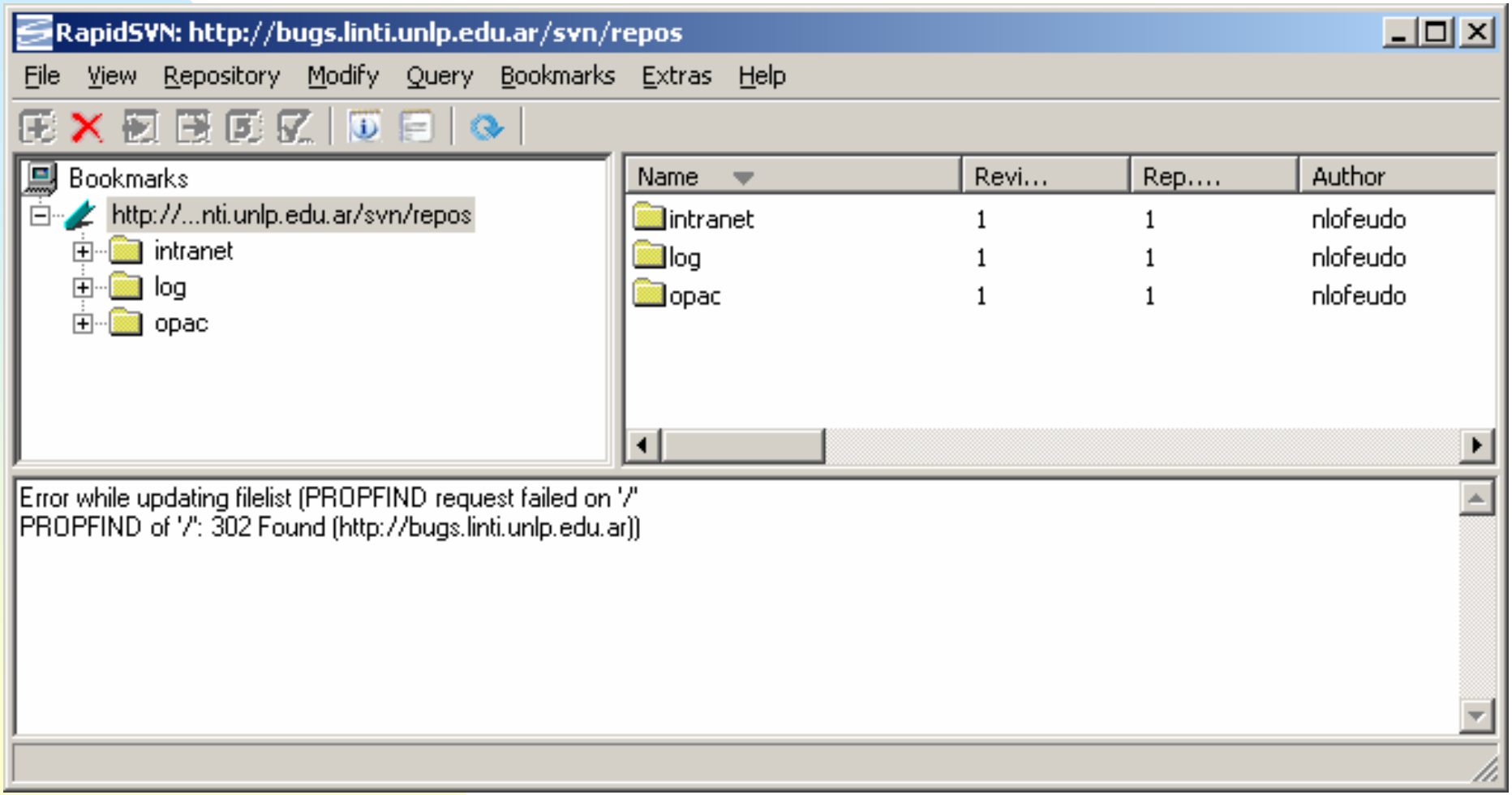
---

r1 | nlofeudo | <fecha hora> (Fri, 28 May 2004) | 1 line
import inicial

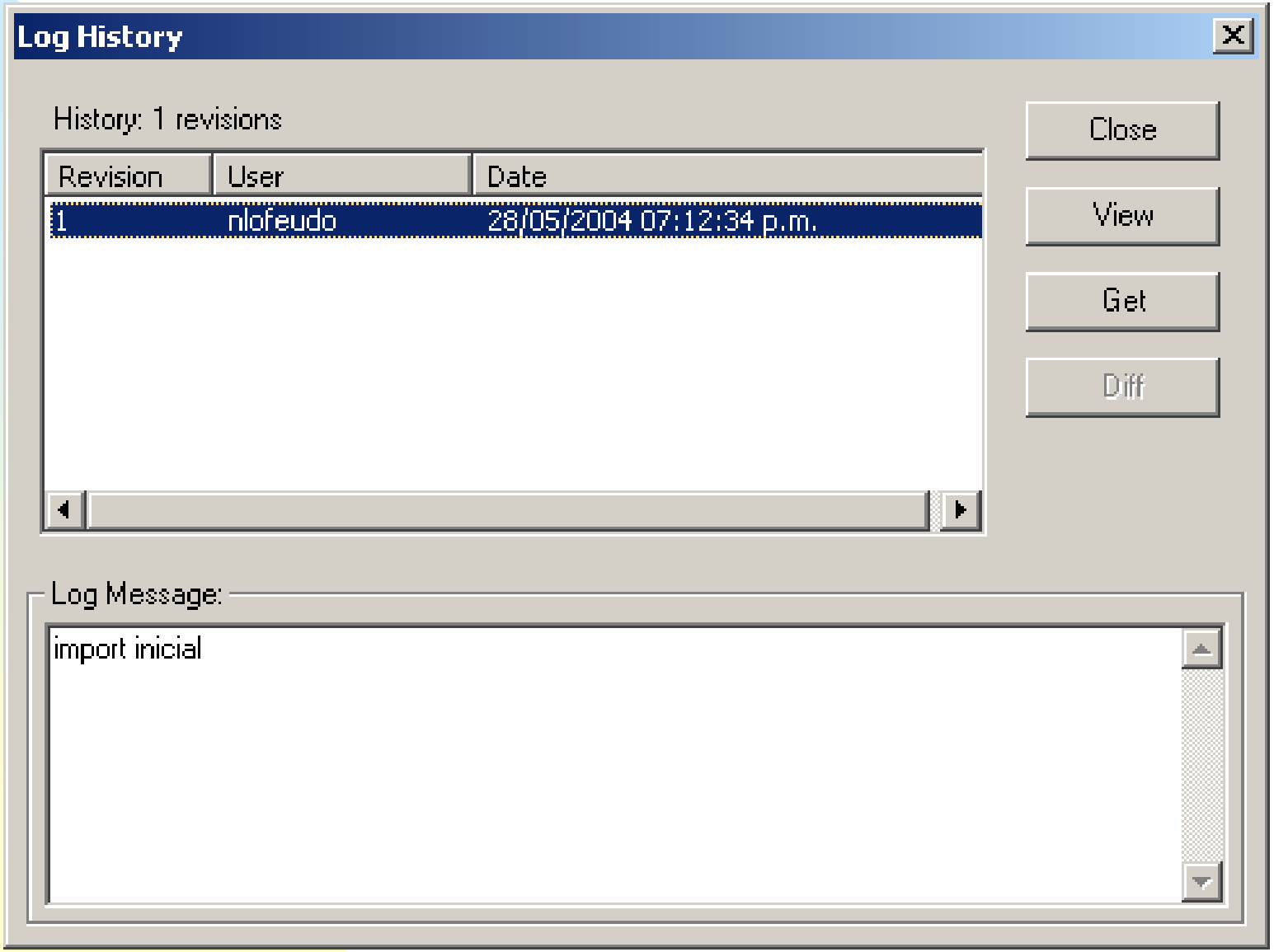
---

r2 | einar | <fecha hora> (Fri, 28 May 2004) | 1 line
Corregido bug #2

Clientes gráficos: RapidSVN



RapidSVN

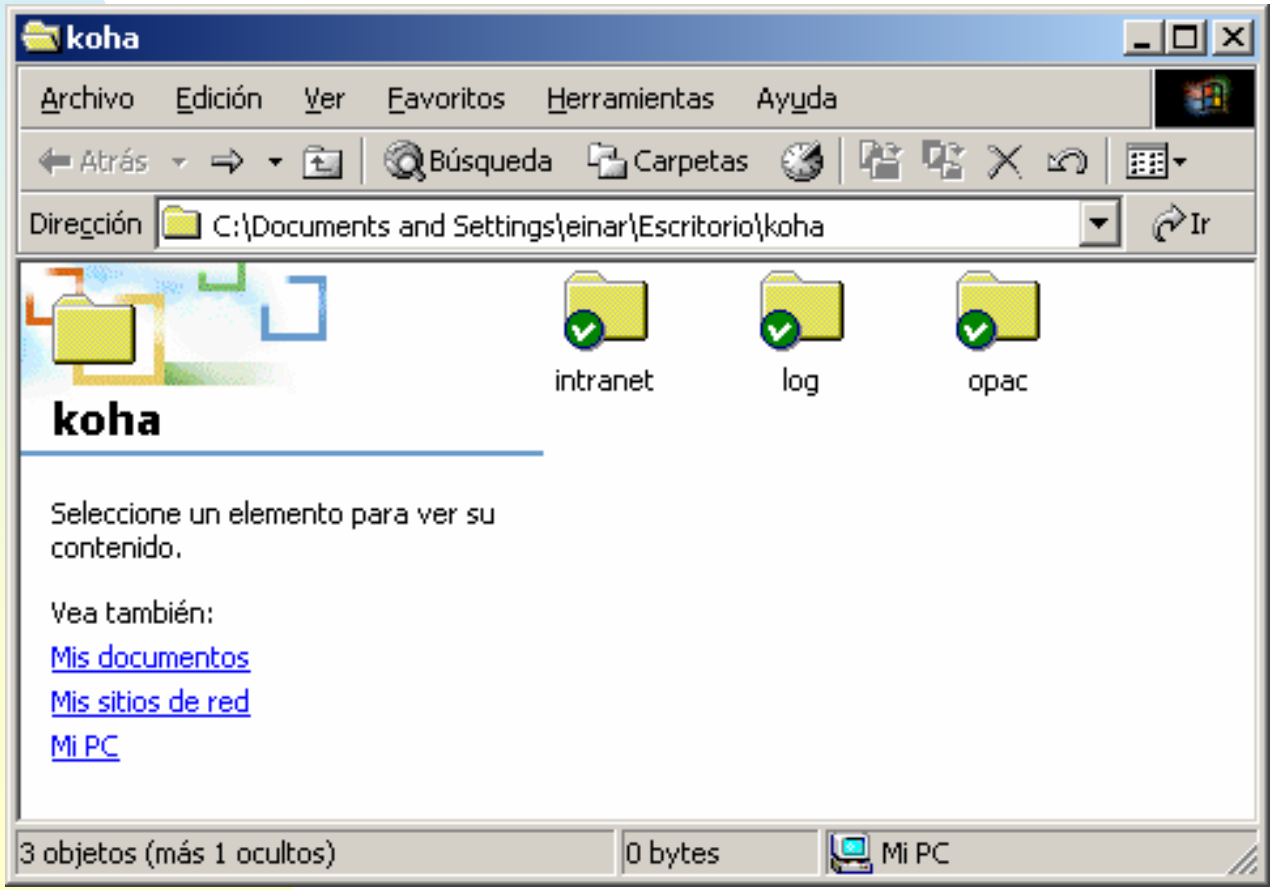

Log History
History: 1 revisions
Revision
User
Date
1
nlofeudo
28/05/2004 07:12:34 p.m.
Close
View
Get
Diff
Log Message:
import inicial

RapidSVN

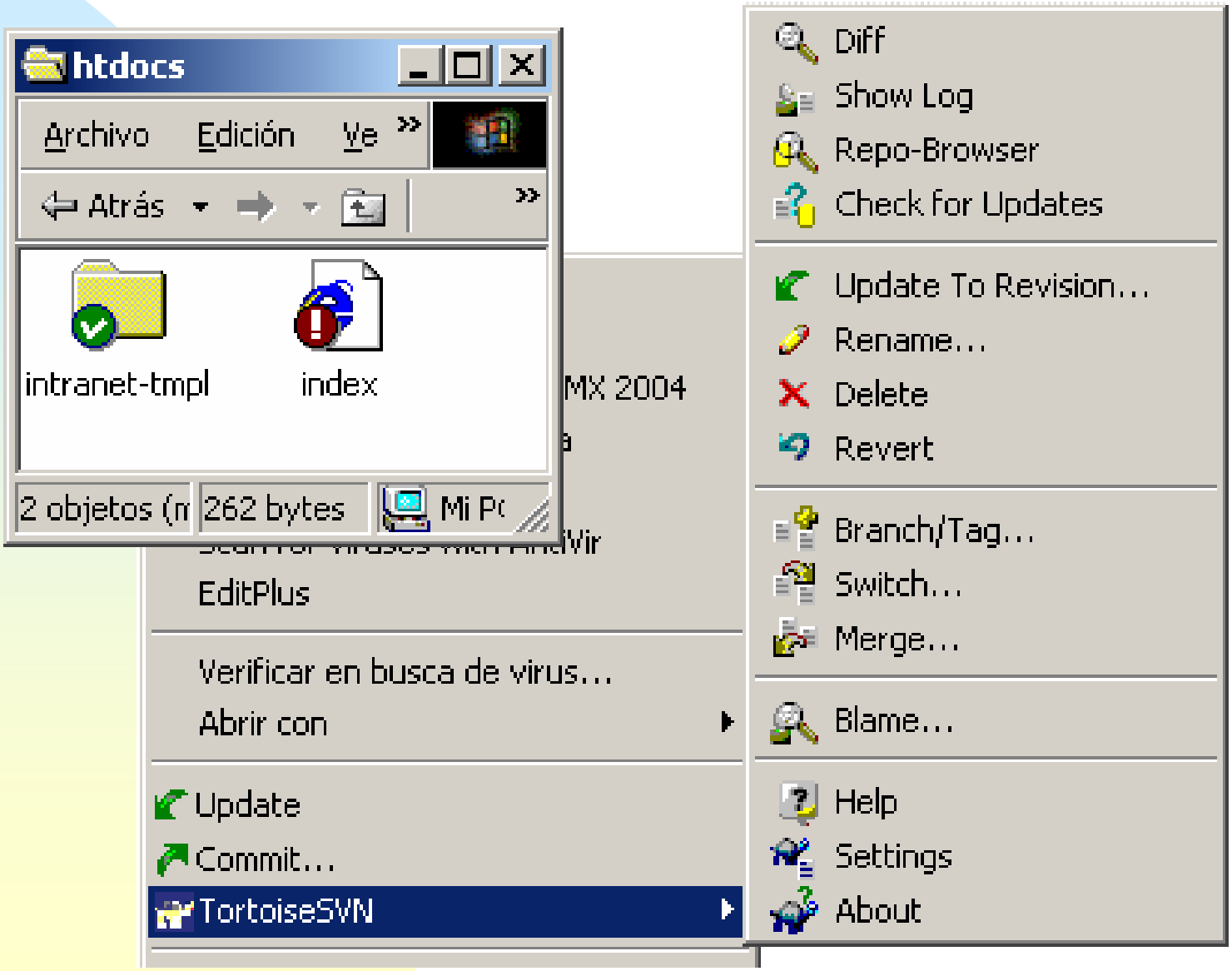
- Interfaz gráfica
- Multiplataforma:
  - Windows
  - GNU/Linux
  - Mac OS 9.x
  - Mac OS X
- Soporta todas las operaciones de SVN

TortoiseSVN

- Integración con el shell de windows
- Marca los archivos/carpetas



TortoiseSVN



Actualización mediante SVN

- ***Para la versión actual se requieren los siguientes módulos extras:***
  - Extutils::Autoinstall
  - Perl-LDAP
  - Convert-ASN1

  El orden de instalación es importante!!!
- ***Y los siguientes rpms para el SVN:***
  - ***apr-0.9.5-0.2.i386.rpm***
  - ***apr-util-0.9.5-0.1.i386.rpm***
  - ***Neon-0.24.6-1.i386.rpm***
  - ***subversion-1.0.5-1.rh80.i386.rpm***

Actualización mediante SVN

- ***Para realizar la actualización:***
  - ***Posicionarse en el directorio /usr/local***
    - ***cd /usr/local***
  - ***Hacer el checkout***
    - ***svn co http://bugs.linti.unlp.edu.ar/svn/reposkoha***

PERL

- ***Introducción***
- ***¿Cómo esta escrito Koha?***
- ***Ejemplos en Koha***

Proyecto KOHA

Grupo de desarrollo UNLP

Introducción: Objetivos

- Definir el concepto de módulo, su alcance y utilización
- Introducir elementos de POO en Perl
- Mostrar ejemplos de uso de módulos típicos
- Crear nuevos módulos
- Expandir las capacidades de la herramienta de programación (Instalación de módulos existentes)

Módulos Perl

- Formato de librería de código
  - Reusabilidad
  - Claridad
  - Simpleza
  - Extensibilidad
- Características de OO al lenguaje
- Identificados externamente con la extensión `.pm`
- Identificados internamente con la palabra reservada package <Nombre>;
  - `package Matematico;`

Módulos Perl

- Hay dos formas de poner disponible los un módulo para los programas.
- 1) Exportando Símbolos
- 2) Mediante llamadas a Métodos (OO)

Módulos Perl

- Ejemplo

```
package Matematico;
require Exporter;
our @ISA = qw(Exporter);
our @EXPORT = qw(suma);

sub suma
{
($a,$b) = @_;
$res= $a + $b;
return($res);
}

sub producto
{
my ($a,$b) = @_;
return($a*$b);
}

1;
```

Comienzo

Exporta símbolo

Procedimientos
Funciones

Variables

Fin

Módulos Perl

```
#!/usr/bin/perl

use Matematico;

 print suma(2,8)."\n";
```

Salida

```
10
```

Referencias

- ***Variables usadas como punteros***
  - ***Ejemplo #1 - Referencia sobre una variable***

```
$nombre = "gabriel";
#Establecer una referencia
$ref = \$nombre;
#Mostrar valores
print "El valor de \$nombre es $nombre \n";
print "El valor de \$ref    es $ref    \n";
print "El valor de la variable referenciada es $$ref \n";
```

- ***Salida***

```
El valor de $nombre es gabriel
El valor de $ref    es SCALAR(0x1c256f0)
El valor de la variable referenciada es gabriel
```

Referencias

- Ejemplo #2 - Referencia sobre un arreglo

```
@mascotas = ("perro", "gato", "canario", "chancho");
#Establecer una referencia para el arreglo
$ref = \@mascotas;
#Mostrar valores
print "El valor de  \$ref   es $ref \n";
print  "El  valor  del  primer   elemento  del  arreglo  es
  $$ref[0]”;
print  "El  valor  del  segundo  elemento  del  arreglo  es
  $$ref[1]";
```

- ***Salida***

```
El valor de  $ref    es ARRAY(0x1c231bc)
El valor del primer  elemento del arreglo es perro
El valor del segundo elemento del arreglo es gato
```

Referencias

- ***Ejemplo #3 - Referencia sobre un hash***
- ***(arreglos que se acceden por clave)***

```
#Asignar un hash con pares atributo=>valor
%mascotas = ('Scooby'=>'perro', 'Silvestre'=>'gato',
  'Tweety'=>'canario');


#Establecer una referencia para el arreglo
$ref = \%mascotas;


#Mostrar valores
print "El valor de  \$ref es $ref \n";
print ”El valor para la clavede la clave \"Scooby\" es
  $$ref{'Scooby'}\n";
```

Salida

```
El valor de $ref es HASH(0x1c231ec)
El valor para la clave "Scooby" es perro
```

Módulos y POO

- Perl no es un lenguaje OO (Orientado a Objetos)
- Solo soporta algunas características (como extensiones al lenguaje)

- Clase: Módulo o paquete con las definiciones de variables y procedimientos.
- Objeto: Referencia a un módulo o paquete utilizada en un programa.
- Atributo: Datos en una variable, un arreglo o hash que forma un objeto. Lo que sería una variable de instancia.
- Método: Subrutina en el módulo o paquete.

Módulos y POO

- Clase → Módulo (package)
- Objeto → Referencia a un package
- Los módulos utilizados en POO requieren de un método constructor (new)

```
use Auto;
$t = new Auto;
```

o

```
$t = new Auto;
```

o

```
$t = Auto::new();
```

Ejemplo: Módulo Auto.pm

- La clase Auto (Ejemplo de POO)

```
#!/usr/bin/perl
package Auto;


sub new  {
 #Obtiene un referencia anónima a un hash
  my($esteObj) = {};
  my ($mar, $col, $pre) = @_;
  $esteObj->{marca}  = $mar;
  $esteObj->{color}  = $col;
  $esteObj->{precio} = $pre;
  return $esteObj;
}
```

Módulo Auto.pm (cont.)

```
sub verPropiedades {
        my($esteObj) = @_;
        print "$esteObj->{marca}   \t";
        print "$esteObj->{color}   \t";
        print "$esteObj->{precio}  \n";
}

sub asignarValor
{
        my($esteObj, $propiedad, $valor) = @_;
        $esteObj->{$propiedad} = $valor;
}
1;
```

Módulo Auto.pm (cont.)

- Uso del módulo Auto.pm

```
#!/usr/bin/perl
use Auto;

$t = new Auto("Renault", "Verde", 10000);
print "Auto t: $t->verPropiedades”;

$r = new Auto("Mercedes", "Plata", 231000);
print "Auto r: $r->verPropiedades;
$t->asignarValor("marca", "Chevrolet");
$r->asignarValor("precio", 553000);

print "Auto r: $r->verPropiedades;
```

Herencia: AutoDeCarrera.pm

- La clase AutoDeCarrera.pm derivada de Auto

```
#!/usr/bin/perl
package AutoDeCarrera;
require Auto;

@ISA = "Auto";

sub new {
        my $este = shift;
        $clase = {};
        bless($clase, $este);
    return($clase);
}
```

Herencia: AutoDeCarrera.pm

- Métodos nuevos

```
sub hacerRuido
{
  print "rrr, rrr, rrrrrr!\n";
}

sub hacerMuchoRuido
{
  print "rrr, rrr, rrrrrr!\t";
  print "rrr, rrr, rrrrrr!\t";
  print "rrr, rrr, rrrrrr!\n";
}
1;
```

Tipos de Módulos

- Pragmáticos (*pragmas*)
  - Operan como directivas de compilador
  - Afectan el comportamiento del programa
    - `use <módulo>`
    - `no <módulo>`
  - Ejemplos

strict	Exige declaraciones
integer	Realiza cálculos enteros (en vez de doble)
constant	Declara constantes
diagnostics	Fuerza diagnósticos en modo debug

Tipos de Módulos

- ***Estándard***
  - ***Permiten extender la funcionalidad del lenguaje***
  - ***Ejemplos***

CGI	***Implementa manejo de CGI***
file	***Manejo de archivos (copia, comparación, path)***
IO	***Front-end a otros módulos de IO (por ejemplo, IO::Socket)***
Math::Trig	***Funciones trigonométricas***
Shell	***Permite correr comandos del SO de forma transparente***

Repositorio de Módulos

- ***CPAN - Comprehensive Perl Archive Network***

  www.cpan.org
  ftp://cpan.if.usp.br/pub/mirror/CPAN/ ***Brasil***
  ftp://sunsite.dcc.uchile.cl/pub/Lang/PERL/ ***Chile***

  - ***Módulos*** ***(En general, código fuente → Se requiere un compilador C)***
  - ***Documentación***
  - ***Guías de estilo***
  - ***Tips***
  - ***Distribuciones***

Instalación de Módulos CPAN

- En general, el proceso completo requiere:
  - Descomprimir el archivo (gzip, zip)
  - Desempaquetar (tar)
  - Compilar (Build) - En Unix
    - `perl MakeFile.PL`
    - `make`
    - `make test`
  - Instalar (Build) - En Unix
    - `make install`

Creación de Nuevos Módulos

- ***Diseño de clases***
  - ***Una clase es un package***
  - ***Un objeto es una referencia (blessed)***
  - ***Un método es una subrutina***
  - ***Una propiedad es una estructura de datos (generalmente, un hash)***
- ***Herencia***
  - ***@ISA***
- ***Constructor***
  - ***new***

Uso de Módulos

- Perl dispone de la directiva `use <nombre>`
- El intérprete “busca”en los directorios listados en @INC

```
use File;                  # Usar el módulo
                                 File.pm
```

- El :: indica subdirectorio

```
use File::Basename;   # Usar el módulo

    File/Basename.pm
```

Koha: Estructura

- Cómo esta escrito Koha?

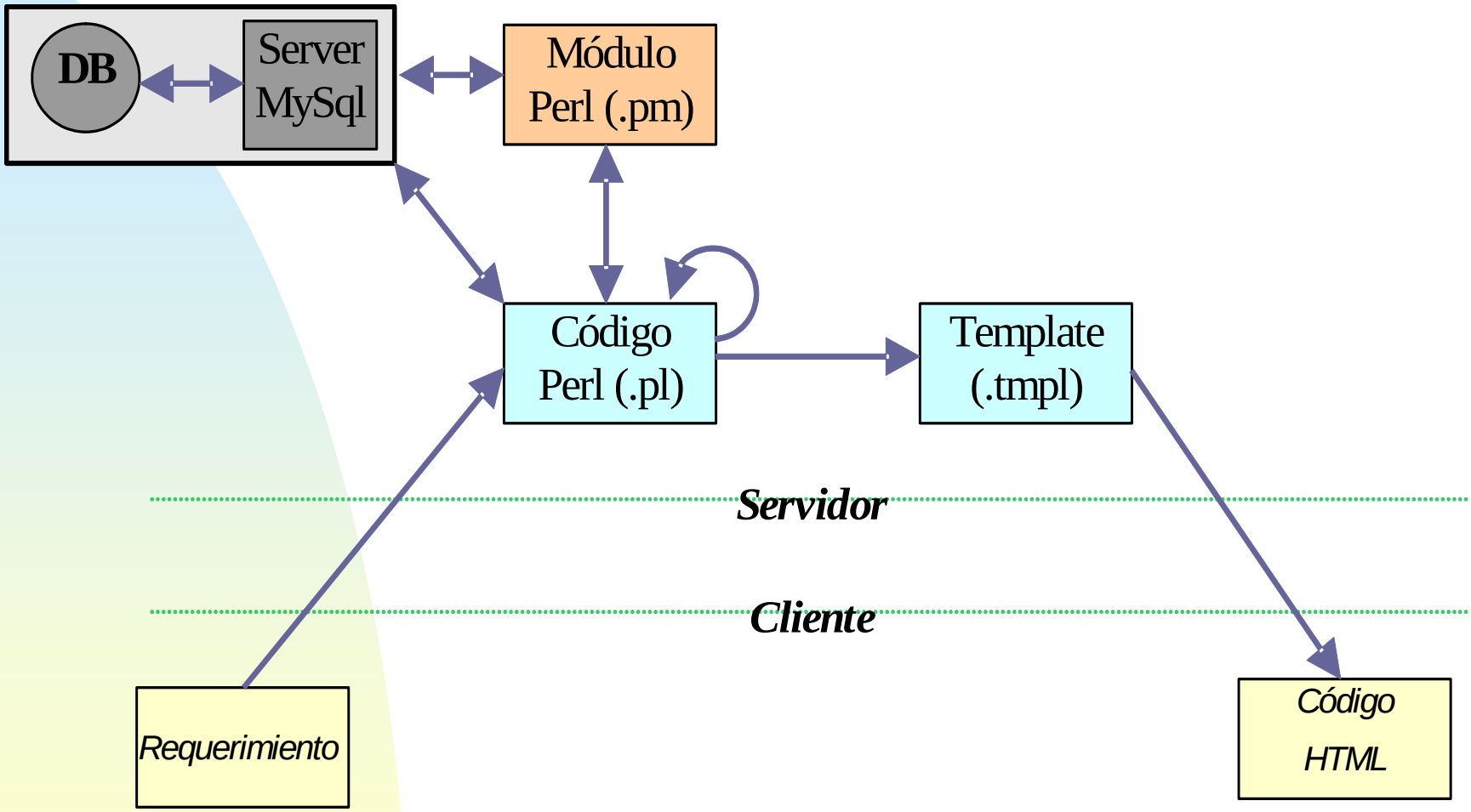
Tres tipos de Archivos:

- ***Los scripts perl (.pl) forman el Núcleo de Koha.***
- ***Los templates (.tmpl) tienen la forma del html que se retorna al cliente.***
- Los módulos perl (.pm) reunen funcionalidad común a diversas partes del sistema.

Koha: Funcionamiento

- ¿Cómo funciona Koha?
- Los usuario invocan a los scripts .pl a través del webserver, y contestan esas invocaciones apoyándose en las funciones que les proveen los .pm y se muestran al usuario mediante los .tmpl.
- En el siguiente gráfico se ilustra esta interoperabilidad:

Koha: Funcionamiento

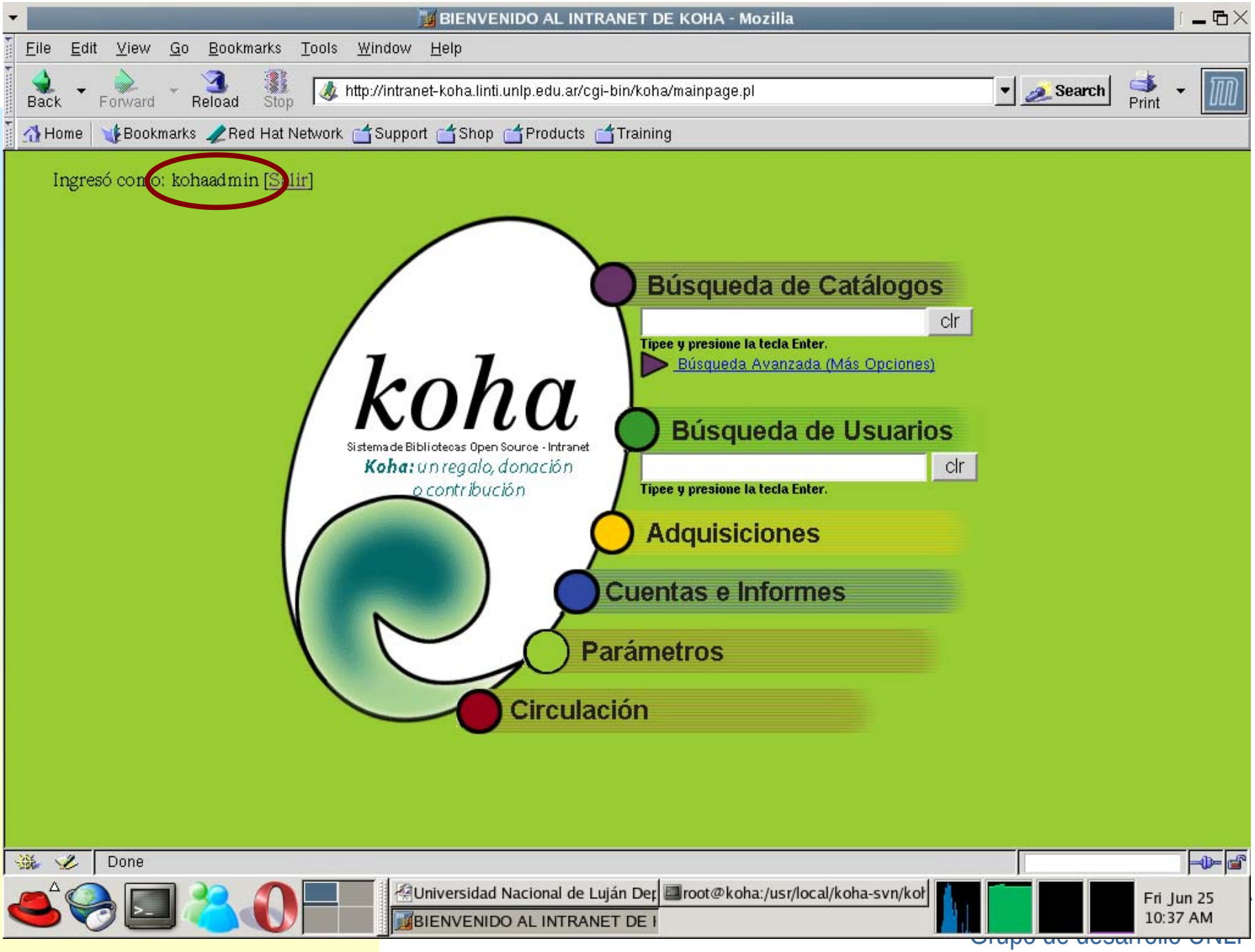


Koha: Funcionamiento

En las siguientes diapositivas se muestra este esquema de funcionamiento

Proyecto KOHA

Grupo de desarrollo UNLP


BIENVENIDO AL INTRANET DE KOHA - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://intranet-koha.linti.unlp.edu.ar/cgi-bin/koha/mainpage.pl
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training
Ingresó como: kohaadmin [Salir]
koha
Sistema de Bibliotecas Open Source - Intranet
Koha: un regalo, donación o contribución
Búsqueda de Catálogos
clr
Tipee y presione la tecla Enter.
Búsqueda Avanzada (Más Opciones)
Búsqueda de Usuarios
clr
Tipee y presione la tecla Enter.
Adquisiciones
Cuentas e Informes
Parámetros
Circulación
Done
Universidad Nacional de Luján Dep
root@koha:/usr/local/koha-svn/koh
BIENVENIDO AL INTRANET DE I
Fri Jun 25
10:37 AM

Código del ejemplo (mainpage.pl)

```
#!/usr/bin/perl
use HTML::Template;
use strict;
require Exporter;
use C4::Database;
use C4::Output;  # contains gettemplate
use C4::Interface::CGI::Output;
use CGI;
use C4::Auth;
my $query = new CGI;
my ($template, $loggedinuser, $cookie)
    = get_template_and_user({template_name => "intranet-main.tmpl",
                        query => $query,  type => "intranet",  authnotrequired => 0,
                        flagsrequired => {catalogue => 1, circulate => 1,  parameters => 1, borrowers =>
1,
                                          permissions =>1, reserveforothers=>1,  borrow => 1,
reserveforself => 1,
                                          editcatalogue => 1, updatecharges => 1, },
                        debug => 1,
                        });
my $marc_p = C4::Context->boolean_preference("marc");
$template->param(NOTMARC => !$marc_p);
output_html_with_http_headers $query, $cookie, $template->output;
```

Código del ejemplo (intranet-main.tmpl)

Es una pequeña parte del archivo. Fue agragado para permitir al usuario hacer el logout desde la página principal. Imprime el nombre de usuario logueado:

```
 <p align=left > Ingres&oacute; como: <!-- TMPL_VAR NAME="loggedinusername" -->  [<a
href=/cgi-bin/koha/mainpage.pl?logout.x=1>Salir</a>]</p>
```

Código del ejemplo (Auth.pm)

```
Sub get_template_and_user {
        my $in = shift;
        my $template = gettemplate($in->{'template_name'}, $in->{'type'});
        my ($user, $cookie, $sessionID, $flags)
                = checkauth($in->{'query'}, $in->{'authnotrequired'}, $in->{'flagsrequired'}, $in->{'type'});

        my $borrowernumber;
        if ($user) {
                $template->param(loggedinusername => $user);
                $template->param(sessionID => $sessionID);

                $borrowernumber = getborrowernumber($user);
                my ($borr, $flags) = getpatroninformation(undef, $borrowernumber);
                my @bordat;
                $bordat[0] = $borr;

                $template->param(USER_INFO => \@bordat);
        }
        return ($template, $borrowernumber, $cookie);
}
```

Funciones Adicionales

- Seguridad
- LDAP
- Integración con SIU-Guaraní

Seguridad: Encriptación de la contraseña

- El usuario hace el request de la página de login y el servidor envía un número aleatorio dentro del cuerpo del html.
- Utilizando código javascript se hace un hash en md5 de la contraseña ingresada por el usuario de la siguiente forma: md5(md5(contraseña)+nroAleatorio)

Seguridad: Encriptación de la contraseña

- Se hace el submit de la contraseña encriptada, junto con el ID, y el número random.
- Del lado del servidor se hace el mismo procedimiento y luego se compara contra lo enviado por el cliente.

Utilización de LDAP:

- Requiere que:
- ***Los usuarios estén cargados en una base de datos LDAP.***
- ***Cada usuario en el LDAP tenga su correspondiente usuario en la base de datos de koha con el mismo uid.***
- ***Configurar las siguientes variables de sistema: ldapenabled (yes, no), ldapinfos (ej: dc=econo) , ldappass (ej: pepe), ldaproot (ej: root) y ldapserver (ej: koha.redes.info.unlp.edu.ar)***

Koha con LDAP: Funcionamiento

- Cuando el usuario hace el requerimiento al servidor, éste se conecta al servidor LDAP e intenta hacer un bind. Si tiene éxito se considera al usuario autenticado
- Luego, se recuperan los datos del usuario en la base de datos de koha a partir del uid y si existen queda terminado el proceso

Koha con SIU-Guaraní

- Se puede configurar a Koha para que sea solo lector de los datos obtenidos de SIU-Guaraní o que tenga la posibilidad de modificar los datos de sus usuarios (por defecto).
- Para evitar que Koha modifique los datos de los usuarios debe setear la variable de sistema ChekUpdateDataEnabled=no.

Koha con SIU-Guaraní

- Se puede configurar a Koha para que sea solo lector de los datos obtenidos de SIU-Guaraní o que tenga la posibilidad de modificar los datos de sus usuarios (por defecto).
- Para evitar que Koha modifique los datos de los usuarios debe setear la variable de sistema ChekUpdateDataEnabled=no.

Modificaciones

- Adición de Funcionalidades
- Solución de Bugs

Proyecto KOHA

Grupo de desarrollo UNLP

Adición de Funcionalidades

Hoy estamos utilizando la siguiente metodología:

- Crear los directorios que alojarán los nuevos módulos:
  - ***El que contenga los archivos perl (.pl)***
  - ***El que contenga los los templates (.tmpl)***

Adición de Funcionalidades (cont)

- ***Programar los nuevos módulos en archivos que se colocarán dentro de los directorios antes creados***:
  - Por ejemplo:
    - ***Los .pl dentro de***

      /usr/local/koha/intranet/cgi-bin/nuevomodulo/
    - ***Los .tmpl dentro de***

/usr/local/koha/intranet/htdocs/intranet-tmpl/default/en/nuevomodulo/

Adición de Funcionalidades (cont)

- ***Si se crea un nuevo modulo .pm para que sea utilizado por nuestros .pl, guardarlo dentro del directorio que contiene los módulos C4 en el directorio AR.***

Por ejemplo:

/usr/local/koha/intranet/modules/C4/AR/

La forma de invocar el módulo Nuevomodulo desde nuestros .pl es mediante la sentencia:

```
use C4::AR::Nuevomodulo;
```

Adición de Funcionalidades (cont)

Manejo de nuevos tmpl:

- **Sugerencia:** Cuando se crea un nuevo template (.tmpl) se recomienda terminar todo nombre de variable utilizado con mayúscula (por ej.: edicioN) para que estas puedan ser diferenciadas del simple texto que se va a mostrar.

  Esto es muy importante para simplificar el uso de herramientas automáticas de traduccón.

Adición de Funcionalidades (cont)

- ***Agregar un link dentro de KOHA apuntando al .pl inicial de nuestro nuevo modulo.***

Por ejemplo:

<a href="/cgi-bin/koha/nuevomodulo/nuevomodulo.pl?parametrosiniciales>Nuevo Módulo<a>

De esta forma, la única diferencia con el proyecto KOHA original es la adición de un link que nos permite acceder a nuestro módulo. Esto facilitará la futura integración con versiones venideras.

Manejo de Errores

- ***Pasos para la resolución de un error:***
  - ***Reportarlo en MANTIS***
  - ***Demarcar del ámbito del error***
  - ***Esta es el área dentro de la cual se “espera” encontrarlo (Los .pl, .pm , .tmpl involucrados)***

IMPORTANTE: Ver los errores en el navegador

Agregar al archivo CGI.pm la línea:

use CGI::Carp 'fatalsToBrowser';

Sin esta línea sólo vemos en el navegador “500 Internal Error Server”

Manejo de Errores (cont)

- ***Antes de buscar el origen del error, conviene dar un vistazo al servicio CVS de Sourceforge :***

http://cvs.sourceforge.net/viewcvs.py/koha

Para ver si hay alguna versión mas nueva de los archivos involucrados en la que ya este solucionado el error.

Manejo de Errores (cont)

- ***Ubicación del error dentro de su ámbito***
- ***Resolución del error en forma clara y siempre comentando las decisiones tomadas.***
- ***Marcarlo como solucionado en el MANTIS Realizando un breve resumen de la solución para que la misma pueda repetirse en caso de ser necesario.***

Manejo de Errores (cont)

- *Los errores más comunes son:*
  - *Entre la pagina y el .PL:*
    - *Envío de datos incorrectos.*
    - *Omisión de algún parámetro.*
  - *Dentro del código Perl (archivos .PL y .PM):*
    - *Variables no inicializadas.*
    - *Sentencias SQL malformadas (muy usual).*
    - *Limites de loops mal definidos.*
    - *Pasaje de parámetros incorrectos.*
    - *Etc.*

Ejemplo de solución de un error

- Busqueda por Materia (subject):
  - Este error fue solucionado recientemente por la comunidad y es conocido como el **bug 752** dentro del proyecto KOHA en www.sourceforge.net**.**
  - ***SÍNTOMAS:Al realizar una búsqueda por Materia (Subject) se produce un error : 404 Not Found.***

Ejemplo de solución de un error (cont)

- ***BÚSQUEDA DEL ERROR:***
  - ***Se comienza en search.pl donde se invoca a la función catalogsearch() del módulo Search.pm.***
  - ***Esta invoca a CatSearch() donde se arma la sentencia SQL y se ejecuta.***
  - ***Los resultados vuelven a search.pl,el cual los pasa al template subject.tmpl para que sean mostrados.***

Ejemplo de solución de un error (cont)

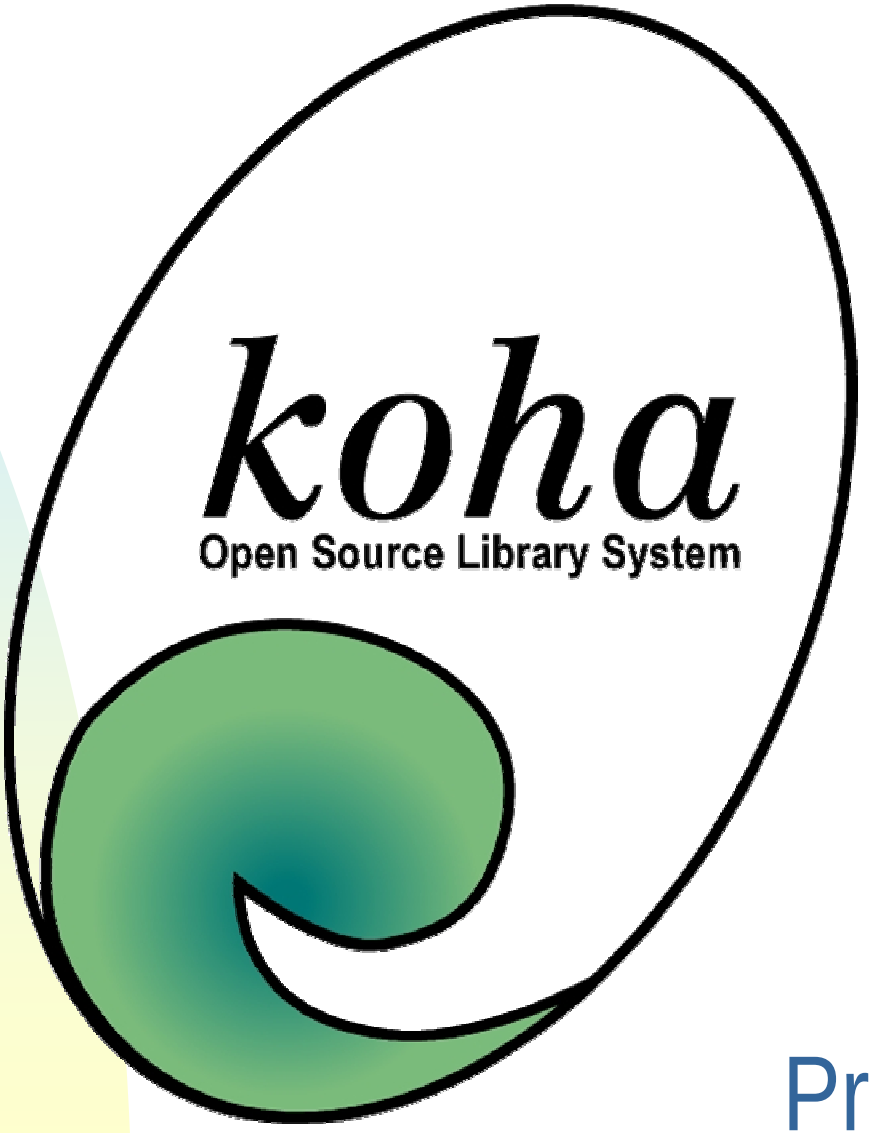
- ***SOLUCIÓN:***
  - *El Servidor MySql nunca realiza la consulta a causa de una sentencia SQL mal formada:*
    - ***select * from bibliosubject, biblioitems where (bibliosubject.biblionumber = biblioitems.biblionumber) and (subject like 'matematicas%' or subject like '% matematicas%' or subject like '%(matematicas)%')***
    - ***order by subject group by subject;***
  - *Linea 1229 del módulo Search.pm :*
    - *$query .= "order by subject group by subject ";*
    - *por*
    - *$query .= "group by subject order by subject ";*
  - *Con lo cual queda:*
    - ***select * from bibliosubject, biblioitems where (bibliosubject.biblionumber = biblioitems.biblionumber) and (subject like 'matematicas%' or subject like '% matematicas%' or subject like '%(matematicas)%')***
    - ***group by subject order by subject;***

Ver los errores en el navegador

Agregar al archivo CGI.pm la línea:

use CGI::Carp 'fatalsToBrowser';

Sin esta línea sólo vemos en el navegador “500 Internal Error Server”



Proyecto KOHA

Grupo de desarrollo UNLP