

UNIVERSIDAD NACIONAL DEL ALTIPLANO
UNIDAD DE POSGRADO FACULTAD DE MECANICA
ELÉCTRICA, ELECTRÓNICA Y SISTEMAS
MAESTRIA EN INGENIERIA DE SISTEMAS



LABORATORIOS

PRESENTADO POR

ING. GOMEZ ALCOS ELMER VALERIO

CURSO

SISTEMAS DISTRIBUIDOS Y PARALELOS

DOCENTE

DR. LENIN HUAYTA FLORES

PUNO – 2025

INDICE

INFORME DE LABORATORIOS	3
INTRODUCCIÓN	3
Laboratorio 01: Rendimiento (Comparación) con Joblib	3
Laboratorio 02: Cliente-Servidor con Sockets	3
Laboratorio 03: RMI en Java	4
Laboratorio 04: Ejemplo con Hilos y Canvas en Java	4
Laboratorio 05: JSP con Apache Tomcat	4
Laboratorio 06: JavaBean en JSP	4
Laboratorio 07: Reserva en JSP	5
Laboratorio 08: Gestión Académica en Django	5
CONSLUSIONES	6
CAPTURAS	7
LABORATORIO 01	7
LABORATORIO 02	8
LABORATORIO 03	9
LABORATORIO 04	11
LABORATORIO 05	12
LABORATORIO 06	13
LABORATORIO 07	14
LABORATORIO 08	16

INFORME DE LABORATORIOS

Curso : Sistemas Distribuidos

Alumno: Elmer Valerio Gómez Alcos

Repositorio de los códigos:

<https://github.com/ValerioGomez/lab-sistemas-distribuidos>

INTRODUCCIÓN

Este informe presenta un resumen detallado de los laboratorios realizados durante el curso de Sistemas Distribuidos. Cada laboratorio abordó aspectos fundamentales de la programación distribuida y concurrente, aplicando tecnologías y herramientas actuales como Python, Java, JSP, Apache Tomcat, Django y paradigmas de concurrencia, con el objetivo de fortalecer las competencias en diseño e implementación de sistemas distribuidos.

Laboratorio 01: Rendimiento (Comparación) con Joblib

Proyecto alojado en: [Google Colab](#)

Descripción:

Se realizó una comparación del rendimiento entre estructuras de datos nativas de Python (listas) y arreglos de NumPy, además de experimentar con paralelización utilizando la librería Joblib.

Logros:

- ✓ Se utilizó la función `time.time()` para medir los tiempos de ejecución con precisión.
- ✓ Se compararon tiempos de procesamiento entre listas nativas y arreglos de NumPy, demostrando que NumPy ofrece un rendimiento superior.
- ✓ Se implementó paralelización con Joblib, variando el número de núcleos y el tamaño del lote de tareas.
- ✓ Se concluyó que la paralelización no siempre mejora el rendimiento, especialmente en operaciones simples o altamente optimizadas como las que realiza NumPy.

Laboratorio 02: Cliente-Servidor con Sockets

Herramienta: NetBeans

Descripción:

Se implementó una aplicación básica de comunicación cliente-servidor utilizando sockets en Java, estableciendo conexión, envío y recepción de mensajes entre ambos extremos.

Logros:

- ✓ Desarrollo e implementación exitosa de un modelo cliente-servidor funcional.
- ✓ Comprensión práctica del manejo de conexiones TCP/IP con sockets en Java.

Laboratorio 03: RMI en Java

Herramienta: NetBeans

Descripción:

Se implementó un sistema distribuido utilizando Remote Method Invocation (RMI) en Java, para invocar métodos remotos y compartir funcionalidad entre aplicaciones cliente y servidor.

Logros:

- ✓ Configuración y desarrollo de una aplicación RMI funcional.
- ✓ Comprensión del uso de interfaces remotas y manejo de objetos distribuidos en Java.

Laboratorio 04: Ejemplo con Hilos y Canvas en Java

Herramienta: NetBeans

Descripción:

Se desarrolló una aplicación gráfica en Java que utilizó hilos (threads) para animar una pelota que rebota dentro de un área de dibujo (Canvas).

Logros:

- ✓ Implementación de concurrencia mediante hilos para animación fluida.
- ✓ Creación de una interfaz gráfica interactiva y visual.

Laboratorio 05: JSP con Apache Tomcat

Descripción:

Se configuró el servidor web Apache Tomcat para desplegar aplicaciones JSP. Se desarrolló y publicó una página JSP que realiza el cálculo de intereses.

Logros:

- ✓ Configuración exitosa del servidor Apache Tomcat en el entorno local.
- ✓ Despliegue y visualización correcta de la página JSP calculoInteres.jsp en el navegador via <http://localhost:8090/JSP01/calculoInteres.jsp>.

Laboratorio 06: JavaBean en JSP

Descripción:

Se integró un JavaBean (TourBean) dentro de una página JSP para manejar y mostrar información dinámica, aplicando el patrón MVC básico.

Logros:

- ✓ Integración funcional de JavaBeans en una aplicación web JSP.
- ✓ Uso de métodos get y set para manipulación dinámica de datos.
- ✓ Configuración y despliegue adecuado en Apache Tomcat.
- ✓ Página accesible vía <http://localhost:8090/JSP02/>.

Laboratorio 07: Reserva en JSP

Descripción:

Desarrollo y despliegue de una aplicación web JSP que simula un sistema de reservas, gestionando adecuadamente solicitudes HTTP y rutas.

Logros:

- ✓ Organización correcta de la estructura del proyecto y configuración del archivo web.xml.
- ✓ Evitación de errores comunes como HTTP 404 mediante manejo adecuado de rutas.
- ✓ Implementación de una página de bienvenida funcional para mejorar la experiencia del usuario.
- ✓ Despliegue exitoso en Apache Tomcat.

Laboratorio 08: Gestión Académica en Django

Descripción:

Desarrollo de una aplicación web utilizando el framework Django para gestionar entidades académicas como alumnos, cursos y matrículas. Se implementaron modelos relacionales y operaciones de persistencia sobre una base de datos relacional, con interfaz administrativa incorporada.

Logros:

- ✓ Creación de un proyecto Django estructurado con entorno virtual y configuración inicial correcta.
- ✓ Desarrollo del modelo de datos con tres entidades clave: Alumno, Curso y Matriculas, incluyendo relaciones y campos representativos.
- ✓ Registro exitoso de la app GestionAcademica en INSTALLED_APPS y configuración de migraciones.
- ✓ Migraciones aplicadas correctamente, creando las tablas en la base de datos.
- ✓ Configuración del panel de administración para gestionar alumnos, cursos y matrículas desde una interfaz amigable.
- ✓ Creación de un súper usuario para acceso administrativo.
- ✓ Despliegue local funcional mediante el servidor de desarrollo de Django.

Tecnologías utilizadas:

- ✓ Python 3
- ✓ Django
- ✓ SQLite (por defecto)
- ✓ Apache Tomcat (solo en laboratorios previos como el 07)
- ✓ Entorno virtual (venv)
- ✓ Pycharm

CONSLUSIONES

1. **Comprensión Integral de Sistemas Distribuidos.** Los laboratorios realizados permitieron adquirir una visión amplia y práctica sobre los conceptos fundamentales de los sistemas distribuidos, incluyendo la programación concurrente, la comunicación a través de redes y el desarrollo de aplicaciones distribuidas con diferentes tecnologías.
2. **Aplicación de Tecnologías Diversas.** El uso de herramientas y lenguajes variados como Python, Java, JSP, Apache Tomcat y Django facilitó el entendimiento de distintos paradigmas y arquitecturas, desde la programación paralela y concurrente hasta el desarrollo web y la gestión de bases de datos.
3. **Experiencia en Programación Concurrente y Paralelismo.** La comparación de rendimiento entre estructuras de datos nativas y NumPy, así como la paralelización con Joblib, evidenció la importancia de elegir adecuadamente las herramientas y técnicas para optimizar el desempeño en sistemas distribuidos.
4. **Dominio de Comunicación Cliente-Servidor.** La implementación práctica de sockets en Java y el desarrollo con RMI reforzaron el conocimiento sobre protocolos TCP/IP y la invocación de métodos remotos, aspectos esenciales para la comunicación eficiente en aplicaciones distribuidas.
5. **Desarrollo de Aplicaciones Web Distribuidas.** La configuración y despliegue de aplicaciones JSP con Apache Tomcat, así como la integración de JavaBeans y la construcción de sistemas web con Django, demostraron la capacidad para crear soluciones distribuidas con interfaces interactivas y gestión dinámica de datos.
6. **Fortalecimiento de Habilidades Prácticas.** El desarrollo de proyectos completos, desde la animación con hilos en Java hasta la gestión académica con Django, contribuyó al fortalecimiento de habilidades técnicas, resolución de problemas y buenas prácticas de programación en entornos distribuidos.
7. **Preparación para Entornos Reales.** Los ejercicios y configuraciones realizadas en los laboratorios simulan escenarios reales de sistemas distribuidos, lo que prepara al alumno para enfrentar desafíos de diseño, implementación y despliegue de aplicaciones distribuidas en entornos profesionales.

CAPTURAS

LABORATORIO 01

The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `colab.research.google.com/drive/1GIXZlhORfLkdJlCe61dPXtxQcdCwkEjg#scrollTo=3ARWzZ6p2pgc`. The notebook title is "Laboratorio 01.ipynb".

The first code cell contains the following Python code:

```
2 start = tm.time()
3 parallel_pool = Parallel(n_jobs=4, batch_size=int(n/4))
4 parallel_sqrt = delayed(sqrt)
5 parallel_tasks = [parallel_sqrt(i) for i in range(n)]
6 parallel_results = parallel_pool(parallel_tasks)
7 end = tm.time()
8 print(f"Tiempo total: {end - start}s")
```

The output of this cell is:

```
Comenzando a calcular ...
Tiempo total: 37.335843086242676s
```

The second code cell contains the following Python code:

```
1 lista = []
2 t1 = tm.time()
3 for i in range(n):
4     lista.append(i)
5 t2 = tm.time()
6 print(t2 - t1)
```

The output of this cell is:

```
0.9895100593566895
```

The bottom of the interface shows the "Variables" and "Terminal" tabs, with the current time at 2:32 p.m. and the Python version 3.

LABORATORIO 02

The screenshot displays the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The title bar indicates the project is 'SocketsClienteServidor' and the IDE version is 'Apache NetBeans IDE 26'. A search bar is located on the right side of the top bar.

The left sidebar shows the 'Projects' view with the following structure:

- SocketsClienteServidor
 - Source Packages
 - cliente
 - Cliente.java
 - servidor
 - Servidor.java
 - Test Packages
 - Libraries
 - Test Libraries

The main editor window displays the code for 'Servidor.java'. The code is as follows:

```
1 package servidor;  
2  
3 import java.io.*;  
4 import java.net.*;  
5 import java.util.logging.Level;  
6 import java.util.logging.Logger;  
7  
8 public class Servidor {  
9     public static void main(String[] args) {  
10         ServerSocket servidor = null;  
11         Socket sc = null;  
12         DataInputStream in;  
13         DataOutputStream out;  
14         final int PUERTO = 9393;  
15  
16         try {  
17             // Se crea el socket del servidor  
18             servidor = new ServerSocket(PUERTO);  
19             System.out.println("Servidor Hola Mundo preparado.");  
20  
21             while (true) {  
22                 // Espera a que un cliente se conecte  
23                 sc = servidor.accept();  
24                 System.out.println("Cliente conectado");  
25             }  
26         }  
27     }  
28 }
```

The 'Output - SocketsClienteServidor (run)' window shows the following log messages:

```
run:  
Servidor Hola Mundo preparado.  
Cliente conectado  
Cliente dice: HolaMundoCliente  
Cliente desconectado
```

The bottom status bar shows the project is running, with a progress indicator and the text 'SocketsClienteServidor (run) running...'. The system tray at the bottom right displays the date and time: '14:01 01/06/2025'.

LABORATORIO 03

RMIClienteServidor - Apache NetBeans IDE 26

Search (Ctrl+I)

391M/5420MB

Projects

- RMIClienteServidor
 - Source Packages
 - Cliente
 - Cliente.java
 - servidor
 - Implementa.java
 - Interfaz.java
 - Servidor.java
 - Test Packages
 - Libraries
 - Test Libraries
 - SocketsClienteServidor

Source

```
9  try {
10      int numPuertoRMI;
11      String nombreNodo;
12
13      InputStreamReader ent = new InputStreamReader(System.in);
14      BufferedReader buf = new BufferedReader(ent);
15
16      System.out.println("Introduce el nombre del nodo del registro RMI:");
17      nombreNodo = buf.readLine();
18
19      System.out.println("Introduce el numero de puerto del registro RMI:");
20      String numPuerto = buf.readLine();
21      numPuertoRMI = Integer.parseInt(numPuerto);
22
23      String URLRegistro = "rmi://" + nombreNodo + ":" + numPuertoRMI + "/holaMundo";
24
25      Interfaz h = (Interfaz) Naming.lookup(URLRegistro);
26      System.out.println("Búsqueda completa");
27
28      String mensaje = h.decirHola("Lenin Huayta Flores");
29      System.out.println("HolaMundoCliente: " + mensaje);
30
31  } catch (IOException | NumberFormatException | NotBoundException e) {
32      System.out.println("Excepción en HolaMundoCliente: " + e);
33  }
```

cliente.Cliente > main > try > URLRegistro

Output - RMIClienteServidor (run-single) ...x

```
run-single:
Introduce el numero del puerto del registro RMI:
9393
El registro RMI no se puede localizar en el puerto 9393
Registro RMI creado en el puerto 9393
Servidor registrado. El registro contiene actualmente:
El registro rmi://localhost:9393 contiene:
//localhost:9393/holaMundo
Servidor Hola Mundo preparado.
```

Explicit type can be replaced with 'var'

RMIClienteServidor (run-single) x 23:33 | INS Windows (CRLF)

14°C
Mayorm. soleado

Buscar

ESP
LAA

14:15
01/06/2025

RMIClienteServidor - Apache NetBeans IDE 26

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

3457/598.0MB

Projects

- RMIClienteServidor
 - Source Packages
 - Cliente
 - Cliente.java
 - servidor
 - Implementa.java
 - Interfaz.java
 - Servidor.java
 - Test Packages
 - Libraries
 - Test Libraries
 - SocketsClienteServidor

Source

```
9      try {
10          int numPuertoRMI;
11          String nombreNodo;
12
13          InputStreamReader ent = new InputStreamReader(System.in);
14          BufferedReader buf = new BufferedReader(ent);
15
16          System.out.println("Introduce el nombre del nodo del registro RMI:");
17          nombreNodo = buf.readLine();
18
19          System.out.println("Introduce el numero de puerto del registro RMI:");
20          String numPuerto = buf.readLine();
21          numPuertoRMI = Integer.parseInt(numPuerto);
22
23          String URLRegistro = "rmi://" + nombreNodo + ":" + numPuertoRMI + "/holaMundo";
24
25          Interfaz h = (Interfaz) Naming.lookup(URLRegistro);
26          System.out.println("Busqueda completa");
27
28          String mensaje = h.decirHola("Valerio Gomez");
29          System.out.println("HolaMundoCliente: " + mensaje);
30
31      } catch (IOException | NumberFormatException | NotBoundException e) {
32          System.out.println("Excepción en HolaMundoCliente: " + e);
33      }
```

cliente.Cliente > main > try > h >

Output

RMIClienteServidor (run-single) x RMIClienteServidor (run) x

```
Introduce el nombre del nodo del registro RMI:
localhost
Introduce el numero de puerto del registro RMI:
9393
Busqueda completa
HolaMundoCliente: Hola Valerio Gomez
BUILD SUCCESSFUL (total time: 6 seconds)
```

Finished building RMIClienteServidor (run).

RMIClienteServidor (run-single) running... x 25:53 INS

14°C Mayorm. soleado

Buscar

ESP LAA 14:16 01/06/2025

LABORATORIO 04

The screenshot shows the Apache NetBeans IDE 26 interface. The main window displays the source code for a Java project named "BounceThread". The code is organized into four files: `Ball.java`, `BounceFrame.java`, `BounceThread.java`, and `BallCanvas.java`. The `Ball.java` file is currently selected, showing the following code:

```
1 package bouncethread;
2
3 import java.awt.Graphics2D;
4 import java.awt.Component;
5 import java.awt.geom.Ellipse2D;
6
7 class Ball {
8     private Component canvas;
9     private static final int XSIZE = 15;
10    private static final int YSIZE = 15;
11    private int x = 0;
12    private int y = 0;
13    private int dx = 2;
14    private int dy = 2;
15
16    public Ball(Component c) {
17        canvas = c;
18    }
19
20    public void draw(Graphics2D g2) {
21        g2.fill(new Ellipse2D.Double(x, y, XSIZE, YSIZE));
22    }
23 }
```

A red rectangle highlights a small window titled "BounceThread" which contains a black dot and "Start" and "Close" buttons. The output window at the bottom shows the command `ant -f "C:\Users\vales\Documents\UNAP\Maestria\TERCER SEMESTRE\SISTEMAS DISTRIBUIDOS Y PARALELOS - GB - SIII\TAREAS\TAREA 02\Laboratorio 04\BounceThread"` and the status "BounceThread (run) running...".

LABORATORIO 05

The screenshot displays the Apache NetBeans IDE environment. The main editor shows a JSP file named `calculoInteres.jsp` with the following code:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%!
double montoPrestamo, tasaInteres;
int cantAños;

// Método para calcular el interés
double calculoInteres(double montoPrestamo, double tasaInteres, int cantAños)
{
    return montoPrestamo * tasaInteres * cantAños / 100;
}
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Cálculo de Interés</title>
</head>
<body>
<h1>Cálculo de Interés</h1>
<%
    montoPrestamo = 1000;
    tasaInteres = 9;
    cantAños = 2;
%>
```

The output window shows the following log messages:

```
Incrementally deploying http://localhost:8090/JSP01
Completed incremental distribution of http://localhost:8090/JSP01
run-deploy:
Browsing: http://localhost:8090/JSP01
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Overlaid on the IDE is a browser window titled "Cálculo de Interés" showing the rendered page. The page displays the following information:

- Monto del Préstamo: 1000.0
- Tasa de Interés: 9.0%
- Cantidad de Años: 2
- Interés Total: 180.0
- Veamos un año más:
Cantidad de Años: 3
Interés Total: 270.0
- Veamos cinco años más:
Cantidad de Años: 15
Interés Total: 1350.0

The system tray at the bottom indicates a temperature of 15°C, the date 01/06/2025, and the time 15:05.

LABORATORIO 06

The screenshot displays the Apache NetBeans IDE environment. The main editor window shows the source code for `TourBean.java`, which implements the `Serializable` interface. The code includes a package declaration, an import statement, a class comment, and methods for constructor, getter, and setter.

```
1 package JavaBean;
2
3 import java.io.Serializable;
4
5 /**
6  * @author PROBOOK
7  */
8 public class TourBean implements Serializable {
9     private String make = "Amantani";
10
11     public TourBean() {
12         // Constructor por defecto
13     }
14
15     public String getMake() {
16         return make;
17     }
18
19     public void setMake(String make) {
20         this.make = make;
21     }
22 }
23
```

The left sidebar shows the project structure for `JSP02`, including `Web Pages`, `META-INF`, `WEB-INF`, `Source Packages`, `Libraries`, and `Configuration Files`.

The bottom output window shows the deployment process:

```
Apache Tomcat or TomEE Log x Apache Tomcat or TomEE x JSP02 (run) x Apache Tomcat or TomEE
Start is in progress...
start?path=/JSP02
OK - Arrancada aplicación en trayectoria de contexto [/JSP02]
run-deploy:
Browsing: http://localhost:8090/JSP02
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

Overlaid on the right is a web browser window titled "Mi primer JavaBean" showing the output of the application:

Usando mi Primer JavaBean

Me gusta la isla de Amantani
Pero también Taquile

The system tray at the bottom indicates a temperature of 15°C, the date 01/06/2025, and the time 15:27.

LABORATORIO 07

The screenshot displays the Apache NetBeans IDE environment. The main editor window shows the code for `web.xml`, which includes the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Quinta Restaurant Arenas del Lago</title>
<style>
p { font-size:11px; font-family:arial; font-weight:bold; }
h5 { font-size:11pt; font-family:verdana; color:#8181F7; }
</style>
<jsp:include page="head.html"/>
</head>
<body bgcolor="#F2F2F2">
<%
// Importar clase JavaBean
javabeans.reserva res = new javabeans.reserva();

// Obtener parámetros del formulario
res.setEmail(request.getParameter("email"));
res.setNombreApe(request.getParameter("nombreApe"));
res.setLocal(request.getParameter("local"));
res.setTelefono(request.getParameter("telefono"));
res.setDomicilio(request.getParameter("domicilio"));
%>
```


The browser window titled "RESTAURANTE EL JUANE" shows the rendered web page. It features a chef icon and the text "RESTAURANTE EL JUANE". Below this, there is a section titled "Eventos" with the text "Puede hacer su reserva eligiendo el mes y el día que desea del año en curso". The form includes a "Mes" dropdown menu set to "Enero", a "Día" input field, and a "Hacer Reserva" button.

The Output window shows the following log messages:

```
Apache Tomcat or TomEE Log x Apache Tomcat or TomEE x Apache Tomcat or TomEE Log x Retriever Output x JSP05 (run) x
compile-jsp:
Incrementally deploying http://localhost:8090/JSP05
Completed incremental distribution of http://localhost:8090/JSP05
run-deploy:
Browsing: http://localhost:8090/JSP05
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

← → ↻ 🏠 ⓘ localhost:8090/JSP05/register.jsp

🗖️ | 📱 Aplicaciones 📧 Gmail 📦 AWS 🔄 GitHub 📘 Facebook 📞 WhatsApp 🇺🇲 UNAP 🌐 SAP Cryst



RESTAURANT ELJUANE

Reservas

Email(*)

Local(*)

Nombre y Apellido (*)

Teléfono de confirmación (*)

Domicilio(*)

Fecha de la Reserva

Hora (*)

Cantidad de Personas (*)

Sector

Comentario

Una Sola Mesa con decoraciõnn por aniversario.

(*) Los campos son obligatorios

← → ↻ 🏠 ⓘ localhost:8090/JSP05/result.jsp

🗖️ | 📱 Aplicaciones 📧 Gmail 📦 AWS 🔄 GitHub 📘 Facebook 📞 WhatsApp 🇺🇲 UNAP 🌐 SAP Cryst



RESTAURANT ELJUANE

Resultado de la Reserva

Email: valerio@gmail.com

Nombre y Apellido: Elmer Valerio Gomez Aclos

Local: Local Norte (Av. Sesquicentenario 1979)

Teléfono: 931537331

Domicilio: Jr.Romulo Diaz Dianderas

Fecha: enero de Abril

Hora: 11:00hs

Cantidad de Personas: 10

Sector: No Fumador

Comentario: Una Sola Mesa con decoraciÃ³nn por aniversario.

Gracias por confiar en nosotros, su reserva ha sido registrada.

LABORATORIO 08

The screenshot displays a development environment with a code editor and a web browser. The code editor shows a Python file named `models.py` with a `class Alumno(models.Model):` definition. The web browser shows the Django admin interface for user management.

Code Editor (models.py):

```
class Alumno(models.Model):  
    SEXOS = (('F', 'Femenino'), ('M', 'Masculino'))  
    ...
```

Web Browser (Django Admin):

The browser shows the Django admin interface for user management. The page title is "Administración de Django". The breadcrumb trail is "Inicio > Autenticación y autorización > Usuarios". The page content is titled "Seleccione usuario a modificar".

Left Sidebar (Navigation):

- AUTENTICACIÓN Y AUTORIZACIÓN
 - Grupos + Añadir
 - Usuarios + Añadir
- GESTIONACADEMICA
 - Alumnos + Añadir
 - Cursos + Añadir
 - Matriculass + Añadir

Right Sidebar (Filters):

- FILTRO
 - Por es staff
 - Todo
 - Sí
 - No
 - Por estado de superusuario
 - Todo
 - Sí
 - No
 - Por activo
 - Todo
 - Sí
 - No
 - Por grupos

Table of Users:

Acción:	NOMBRE DE USUARIO	DIRECCIÓN DE CORREO ELECTRÓNICO	NOME
<input type="checkbox"/>	admin	admin@gmail.com	
<input type="checkbox"/>	valerio	valesitoo@gmail.com	Elmer Valeri

2 usuarios

Taskbar:

The taskbar shows the system clock as 17:15 on 01/06/2025. The language is set to ESP (Spanish).

Shop Ray-Ban Meta AI Glasses xABBA - Dancing Queen (OfiSeleccione curso a modificar | S

127.0.0.1:8000/admin/GestionAcademica/curso/

AplicacionesGmailAWSGitHubFacebookWhatsAppUNAPSAP Crystal Reports...PlatziMapsVSCodeFirebasePlay ConsoleTodos los marcadores

Administración de DjangoBIENVENIDOS, ADMIN. VER EL SITIO / CAMBIAR CONTRASEÑA / CERRAR SESIÓN

Inicio > Gestionacademica > Cursos

Empiece a escribir para filtrar...

AUTENTICACIÓN Y AUTORIZACIÓN

Grupos + Añadir

Usuarios + Añadir

GESTIONACADEMICA

Alumnos + Añadir

Cursos + Añadir

Matriculass + Añadir

Seleccione curso a modificar

AÑADIR CURSO +

Acción: ----- Ir seleccionados 0 de 4

☐ CURSO

☐ SISTEMAS DISTRIBUIDOS (5)

☐ PROGRAMACION (4)

☐ MATEMATICA (4)

☐ FISICA (4)

4 cursos

13°C
Mayorm. soleado

Buscar

ESP
LAA

16:56
01/06/2025