



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 2 - Sistema gestor de Alumnos

Unidad de aprendizaje: Aplicaciones para Comunicaciones de Red

Grupo: 3CM8

Alumnos(a):

Nicolás Sayago Abigail
Ramos Díaz Enrique

Profesor(a):

Moreno Cervantes Axel

05 de Marzo 2019

Contents

1	Introducción	2
2	Desarrollo	2
2.1	Alumno	2
2.2	Catalogo	4
2.3	Grupo	8
2.4	Horario	10
2.5	Materia	12
2.6	Servidor	13
2.6.1	Main	13
2.6.2	void autentificarLogin(cl, dis)	15
2.6.3	void enviarGrupos(cl)	16
2.6.4	void recibirHorario(cl)	16
2.7	Cliente	19
2.7.1	Alumno iniciarSesion(boletaTmp, psswdTemp)	19
2.7.2	void obtenerGrupos()	20
2.7.3	void enviarHorario(grupos, materia, boleta)	21
2.8	Interfaces de Usuario	22
2.8.1	Login	22
2.8.2	MiniMenu	26
2.8.3	Inscribir	30
2.8.4	Ver calificaciones	38
2.8.5	Ver horario	43
3	Pruebas	47
3.1	Servidor	48
3.2	Inicio de sesión	48
3.3	Mini menú	49
3.4	Inscribir Horario	50
3.5	Ver calificaciones	55
3.6	Ver horario	55
4	Posibles mejoras	56
5	Conclusiones	56
5.1	Nicolás Sayago Abigail	56
5.2	Ramos Diaz Enrique	56

1 Introducción

En esta práctica se implementará una aplicación de gestión de alumnos, que permita a un alumno inscribir su horario académico, ver su horario inscrito y ver sus calificaciones.

Se utilizarán sockets de flujo bloqueante para comunicar al cliente con el servidor, y el lenguaje de programación Java.

2 Desarrollo

Para la implementación de ésta práctica se utilizaron las siguientes clases:

- Alumno
- Catalogo
- Grupo
- Horario
- Materia
- Servidor
- Login
- MiniMenu
- CalificacionesV
- HorarioV
- InscribirV

2.1 Alumno

Esta clase define a nuestro objeto : **Alumno**, el cual tiene las siguientes características:

- ✓ **Boleta**: Cada alumno tiene su boleta, y ninguna se repite.
- ✓ **Contraseña**: contraseña asignada.
- ✓ **Nombre**: Nombre del alumno.
- ✓ **Primer apellido**: Primer apellido del alumno
- ✓ **Segundo apellido**: Segundo apellido del alumno
- ✓ **Horario**: Cada alumno debe tener un solo horario.
- ✓ **Inscripción**: Funciona como bandera para saber si un alumno esta inscrito o no.
- ✓ **Foto**: Foto del alumno.

✓ Code

```
import java.io.Serializable;
import javax.swing.ImageIcon;

public class Alumno implements Serializable {
```

```
private int boleta;
private String contrasenia;
private String nombre;
private String primerAp;
private String segundoAp;
private Horario horario;
private boolean inscripcion;
private ImageIcon foto;

public Alumno(int boleta, String contrasenia, String nombre, String
    ↪ primerAp, String segundoAp) {
    this.boleta = boleta;
    this.contrasenia = contrasenia;
    this.nombre = nombre;
    this.primerAp = primerAp;
    this.segundoAp = segundoAp;
}

/*****
/*                               SETTERS
    ↪                               */
*****/

public void setHorario(Horario horario) {
    this.horario = horario;
}

public void setInscripcion(boolean inscripcion) {
    this.inscripcion = inscripcion;
}

public void setFoto(String nombreArchivo) {
    this.foto = new ImageIcon(nombreArchivo);
}

/*****
/*                               GETTERS
    ↪                               */
*****/

public int getBoleta() {
    return this.boleta;
}
```

```
public String getNombreCompleto() {
    return this.nombre + " " + this.primerAp + " " +
        this.segundoAp;
}

public Horario getHorario() {
    return this.horario;
}

public boolean getInscripcion() {
    return this.inscripcion;
}

public ImageIcon getFoto() {
    return this.foto;
}

public String getContrasenia(){
    return this.contrasenia;
}
}
```

2.2 Catalogo

Esta clase nos sirve para cargar los datos del MiniSaes, debido a que no usamos una base de datos, entonces tenemos que agregar datos simulando que se encuentran guardadas. Simulamos tener alumnos, profesores, grupos, materias y horas asignadas para cada una. Esta clase tiene una fuerte participación del lado del **Servidor**.

✓Code

```
import java.util.*;

public class Catalogo {

    // Cargamos toda la informacion para trabajar
    static Materia[] materias = new Materia[10];
    static String[] horas = new String[9];
    static String[] prof = new String[10];
    static Grupo[] grupo = new Grupo[3];
    static Alumno[] alumnos = new Alumno[5];

    public static Materia[] getMaterias() {
        return materias;
    }
}
```

```
}

public static Grupo[] getGrupos() {
    return grupo;
}

public static Alumno[] getAlumnos() {
    return alumnos;
}

public static void cargar() {
    // Se cargan 10 materias
    materias[0] = new Materia(0, "Quimica VI");
    materias[1] = new Materia(1, "Web Application Development");
    materias[2] = new Materia(2, "Instrumentacion");
    materias[3] = new Materia(3, "Ingenieria Software");
    materias[4] = new Materia(4, "Matematicas Avanzadas para la
        ↪ Ingenieria");
    materias[5] = new Materia(5, "Sistemas Operativos");
    materias[6] = new Materia(6, "Tecnologias para la Web");
    materias[7] = new Materia(7, "Analisis de Algoritmos");
    materias[8] = new Materia(8, "Calculo Aplicado");
    materias[9] = new Materia(9, "Aplicaciones para
        ↪ Comunicaciones de Red");

    // Se cargan alumnos
    alumnos[0] = new Alumno(2014081268, "prueba123", "Enrique",
        ↪ "Ramos", "Diaz");
    alumnos[1] = new Alumno(2014081144, "prueba123", "Angel",
        ↪ "Hernandez", "Molina");
    alumnos[2] = new Alumno(2014171285, "prueba123", "Abigail",
        ↪ "Nicolas", "Sayago");
    alumnos[3] = new Alumno(2015674078, "prueba123", "Oribel",
        ↪ "Peralta", "Morones");
    alumnos[4] = new Alumno(2014011111, "prueba123", "Son",
        ↪ "Goku", "");

    alumnos[0].setFoto("./fotos/" + alumnos[0].getBoleta() +
        ↪ ".png");
    alumnos[1].setFoto("./fotos/default.png");
    alumnos[2].setFoto("./fotos/" + alumnos[2].getBoleta() +
        ↪ ".png");
    alumnos[3].setFoto("./fotos/" + alumnos[3].getBoleta() +
        ↪ ".png");
}
```

```
alumnos[4].setFoto("./fotos/" + alumnos[4].getBoleta() +
    ↪ ".png");

alumnos[0].setInscripcion(false);
alumnos[1].setInscripcion(false);
alumnos[2].setInscripcion(false);

// Se cargan las horas matutinas y vespertinas
horas[0] = "7:00 - 8:30";
horas[1] = "8:30 - 10:00";
horas[2] = "10:30 - 12:00";
horas[3] = "12:00 - 13:30";
horas[4] = "13:30 - 15:00";
horas[5] = "15:00 - 16:30";
horas[6] = "16:30 - 18:00";
horas[7] = "18:00 - 19:30";
horas[8] = "19:30 - 21:00";

// Se cargan los profesores para cada materia
prof[0] = "Copca Ramirez Vargas";
prof[1] = "Hermes Francisco Montes Casiano";
prof[2] = "Juan Carlos Tellez Barrera";
prof[3] = "Jose Jaime Lopez Rabadan";
prof[4] = "Ignacio Rios de la Torre";
prof[5] = "Jorge Cortes Galicia";
prof[6] = "Jose Antonio Ramirez";
prof[7] = "Edgardo Adrian Franco Martinez";
prof[8] = "Hector Rojas Luna";
prof[9] = "Axel Ernesto Moreno Cervantes";

grupo[0] = new Grupo(0, "2019-2", "3CM8");
grupo[1] = new Grupo(1, "2019-2", "3CM9");
grupo[2] = new Grupo(2, "2019-2", "3CV8");

for(int i = 0; i < 6; i++) {
    grupo[0].setMaterias(materias[i], i);
    grupo[0].setProfesores(prof[i], i);
}

int z = 0;
for(int i = 2; i < 8; i++) {
    grupo[1].setMaterias(materias[i], z);
    grupo[1].setProfesores(prof[i], z);
}
```

```
        z++;
    }

    z = 0;
    for(int i = 4; i < 10; i++) {
        grupo[2].setMaterias(materias[i], z);
        grupo[2].setProfesores(prof[i], z);
        z++;
    }

    for(int x = 0; x < 2; x++) {
        grupo[x].setHoras(horas[0], 0, 0);
        grupo[x].setHoras(horas[0], 0, 3);
        grupo[x].setHoras(horas[1], 0, 4);

        grupo[x].setHoras(horas[0], 1, 1);
        grupo[x].setHoras(horas[0], 1, 2);
        grupo[x].setHoras(horas[0], 1, 4);

        grupo[x].setHoras(horas[1], 2, 0);
        grupo[x].setHoras(horas[1], 2, 2);
        grupo[x].setHoras(horas[1], 2, 3);

        grupo[x].setHoras(horas[2], 3, 0);
        grupo[x].setHoras(horas[1], 3, 1);
        grupo[x].setHoras(horas[2], 3, 3);

        grupo[x].setHoras(horas[3], 4, 0);
        grupo[x].setHoras(horas[3], 4, 2);
        grupo[x].setHoras(horas[3], 4, 3);
        grupo[x].setHoras(horas[3], 4, 4);

        grupo[x].setHoras(horas[2], 5, 1);
        grupo[x].setHoras(horas[2], 5, 2);
        grupo[x].setHoras(horas[2], 5, 4);
    }

    grupo[2].setHoras(horas[4], 0, 0);
    grupo[2].setHoras(horas[5], 0, 1);
    grupo[2].setHoras(horas[5], 0, 2);
    grupo[2].setHoras(horas[5], 0, 4);

    grupo[2].setHoras(horas[4], 1, 1);
```



```

        grupo[2].setHoras(horas[4], 1, 2);
        grupo[2].setHoras(horas[4], 1, 3);

        grupo[2].setHoras(horas[5], 2, 0);
        grupo[2].setHoras(horas[5], 2, 3);
        grupo[2].setHoras(horas[6], 2, 4);

        grupo[2].setHoras(horas[7], 3, 0);
        grupo[2].setHoras(horas[7], 3, 2);
        grupo[2].setHoras(horas[7], 3, 3);

        grupo[2].setHoras(horas[8], 4, 0);
        grupo[2].setHoras(horas[7], 4, 1);
        grupo[2].setHoras(horas[8], 4, 3);

        grupo[2].setHoras(horas[8], 5, 1);
        grupo[2].setHoras(horas[8], 5, 2);
        grupo[2].setHoras(horas[8], 5, 4);
    }
}

```

2.3 Grupo

Esta clase define a nuestro objeto : **Grupo**, el cual tiene las siguientes características:

- ✓ **Id**: Cada grupo debe tener un identificador.
 - ✓ **Materias**: Los grupos deben tener materias, usamos de 6 a 8 materias por grupo.
 - ✓ **Horas**: Cada grupo debe tener horas definidas para cada materia.
 - ✓ **Periodo**: Cada grupo tiene su respectivo periodo, el periodo va de acuerdo al semestre.
 - ✓ **Nombre**: Un grupo tiene su respectivo nombre. Usamos la nomenclatura que tiene ESCOM.
 - ✓ **Profesores**: Cada grupo tiene profesores y cada materia tiene un profesor.
- ✓ **Code**

```

import java.io.Serializable;

public class Grupo implements Serializable {
    private int Id;
    private Materia[] materias;

```

```
private String[] profesores;
private String[] [] horas;
private String periodo;
private String nombre;

public Grupo(int Id, String periodo, String nombre) {
    this.Id = Id;
    this.periodo = periodo;
    this.nombre = nombre;
    this.materias = new Materia[6];
    this.profesores = new String[6];
    this.horas = new String[6][5]; // 6 materias, 5 dias a la
    ↪ semana

    for(int i = 0; i < 6; i++) {
        for(int j = 0; j < 5; j++) {
            this.horas[i][j] = "";
        }
    }
}

public void setMaterias(Materia materia, int i) {
    this.materias[i] = materia;
}

public void setHoras(String hora, int i, int j) {
    this.horas[i][j] = hora;
}

public void setProfesores(String profesor, int i) {
    this.profesores[i] = profesor;
}

public int getId() {
    return this.Id;
}

public Materia[] getMaterias() {
    return this.materias;
}

public String[] getProfesores() {
    return this.profesores;
}
```

```
    public String[] [] getHoras() {  
        return this.horas;  
    }  
  
    public String getPeriodo() {  
        return this.periodo;  
    }  
  
    public String getNombre() {  
        return this.nombre;  
    }  
}
```

2.4 Horario

Esta clase define a nuestro objeto : **Horario**, y es usado cuando el alumno ya se inscribió en el sistema. El horario tiene las siguientes características:

- ✓ **Numero de materias:** Idealmente, el número de materias debería estar restringido, en este caso puedes registrar las materias que desees.
- ✓ **Materias:** Los grupos deben tener materias, idealmente deben estar restringidos pero en este caso puedes inscribir las materias que quieres.
- ✓ **Grupos:** Un alumno puede tener más de un grupo puesto que no esta limitado a escoger solo de uno. Además, cada materia pertenece a un grupo.
- ✓ **Horas:** Cada grupo debe tener horas definidas para cada materia.
- ✓ **Nombre:** Un grupo tiene su respectivo nombre. Usamos la nomenclatura que tiene ESCOM.
- ✓ **Profesores:** Cada grupo tiene profesores y cada materia tiene un profesor.

✓ Code

```
import java.io.Serializable;  
  
public class Horario implements Serializable {  
    private int numMaterias;  
    /*Solo nos interesan los Ids del grupo*/  
    private Materia[] materias;  
    private Grupo[] grupos;  
    private int[] califs;  
}
```

```

private String[] profesores;
private String[] [] horas;

public Horario(int numMaterias) {
    this.numMaterias = numMaterias;
    this.materias = new Materia[this.numMaterias];
    this.grupos = new Grupo[this.numMaterias];
    this.califs = new int[this.numMaterias];
    this.profesores = new String[this.numMaterias];
    this.horas = new String[this.numMaterias][5]; // N materias,
    ↪ 5 dias a la semana

    for(int i = 0; i < this.numMaterias; i++) {
        for(int j = 0; j < 5; j++) {
            this.horas[i][j] = "";
        }
    }
}

/*****
/*                               SETTERS
↪                               */
/*****/
public void setMaterias(Materia materia, int i) {
    this.materias[i] = materia;
}

public void setHoras(String hora, int i, int j) {
    this.horas[i][j] = hora;
}

public void setProfesores(String profesor, int i) {
    this.profesores[i] = profesor;
}

public void setGrupos(Grupo grupo, int i) {
    this.grupos[i] = grupo;
}

public void setCalifs(int calif, int i) {
    this.califs[i] = calif;
}

/*****/

```

```

/*                                     GETTERS
↪                                     */
/*****/
public Materia[] getMaterias() {
    return materias;
}

public int getNumMaterias() {
    return numMaterias;
}

public Grupo[] getGrupos() {
    return grupos;
}

public int[] getCalifs() {
    return califs;
}

public String[] getProfesores() {
    return this.profesores;
}

public String[][] getHoras() {
    return this.horas;
}
}

```

2.5 Materia

Esta clase define a nuestro objeto : **Materia**, representando a las materias que se pueden inscribir o las que el usuario ya ha inscrito. Una materia es impartida por un profesor y los grupos tienen muchas materias. La materia tiene las siguientes características:

- ✓ **Id**: Cada materia tiene un número identificador.
- ✓ **Nombre**: Cada materia tiene su propio nombre.
- ✓ **Code**

```

import java.io.Serializable;

public class Materia implements Serializable {

```

```
private int Id;
private String nombre;

public Materia(int Id, String nombre) {
    this.Id = Id;
    this.nombre = nombre;
}

public int getId() {
    return this.Id;
}

public String getNombre() {
    return this.nombre;
}
}
```

2.6 Servidor

Ésta clase es la encargada de atender las peticiones del cliente, procesarlas y enviarle las respuestas e información. Básicamente carga los datos, muestra un horario y crea un horario seleccionado.

2.6.1 Main

Aquí es donde todos las peticiones, métodos y respuestas interactúa. Al iniciar el servidor, cargamos todos los datos que se encuentran en la clase **Catalogo**. Se cargan las materias, grupos y alumnos. Después, creamos un ServerSocket, asignándole el mismo puerto que se asigmo en el socket del cliente, y luego creamos un ciclo infinito para estar siempre escuchando a los clientes que se conecten en cualquier momento (creando un socket nuevo de respuesta con el método accept()). También creamos los flujos de entrada y salida del socket. ✓**Code**

```
public static void main(String[] args) {
    // Construimos nuestros catalogos
    Catalogo.cargar();
    materias = Catalogo.getMaterias();
    grupos = Catalogo.getGrupos();
    alumnos = Catalogo.getAlumnos();

    System.out.println("Catalogos creados.");
    try {
        ServerSocket s = new ServerSocket(4321);
```

```

s.setReuseAddress(true);
System.out.println("Servidor Mini SAES iniciado,
    ↪ esperando alumnos/clientes...");

for( ; ; ) {
    Socket cl = s.accept();
    // InputStream
    DataInputStream dis = new
        ↪ DataInputStream(cl.getInputStream());
    System.out.println("\n\nCliente conectado
        ↪ desde " + cl.getInetAddress() + " " +
        ↪ cl.getPort());

    int bandera = dis.readInt();
    // Login
    if(bandera == 0)
        autentificarLogin(cl, dis);

    // Enviar grupos con materias
    else if(bandera == 1)
        enviarGrupos(cl);

    // Recibir grupos y materias para horario
    else if(bandera == 2)
        recibirHorario(cl, dis);

    else
        System.out.println("Error al atender
            ↪ la solicitud del cliente.");

    dis.close();
    cl.close();
} // Fin for
} // Fin try
catch(Exception e) {
    e.printStackTrace();
} // Fin catch
}

```

Ahora bien, leemos una bandera de tipo entero, que nos ayuda a regular las acciones que tomará el servidor según la acción que se realizó en el cliente MiniSaes:

- **La bandera vale 0:** El cliente inicia sesión desde su equipo, el servidor debe recibir el número de boleta y contraseña para autentificarlos.

- **La bandera vale 1:** El servidor enviará grupos con las materias correspondientes.
- **La bandera vale 2:** El servidor recibe grupos y materias que el cliente creo desde la interfaz de usuario.

2.6.2 void autentificarLogin(cl, dis)

El servidor recibe el número de boleta y la contraseña del cliente. Se busca en el catalogo de alumnos el número de boleta que coincida con alguno y si existe entonces enviamos el objeto alumno al cliente.

✓ Code

```
public static void autentificarLogin(Socket cl, DataInputStream dis) {
    try {
        // Envia y recibe objetos
        int boleta_temp = dis.readInt();
        String passwd_temp = dis.readUTF();
        System.out.println("Datos recibidos: " + boleta_temp + " " +
            ↳ passwd_temp + ". Buscando...");

        boolean existe = false;
        int numReg = 0;
        Alumno alumnoActual = null;

        for(int i = 0; i < alumnos.length; i++) {
            int b = alumnos[i].getBoleta();
            String p = alumnos[i].getContrasenia();

            // Para el inicio de sesión
            if(boleta_temp == b && passwd_temp.equals(p)) {
                existe = true;
                numReg = i;
            }
        }

        if(existe) {
            alumnoActual = alumnos[numReg];
            System.out.println("Objeto alumno enviado con
                ↳ boleta: " + alumnoActual.getBoleta());
        }
        else {
            System.out.println("Alumno no encontrado, enviando
                ↳ null...");
        }
    }
}
```



```

        ObjectOutputStream oos = new
            ↳ ObjectOutputStream(cl.getOutputStream());
        oos.writeObject(alumnoActual);
        oos.flush();
        oos.close();
        // Limpiamos memoria
        alumnoActual = null;
    }
    catch(Exception e) {
        e.printStackTrace();
    } // Fin catch
}

```

2.6.3 void enviarGrupos(cl)

El servidor envía los grupos disponibles en el catalogo para ser mostrados en la interfaz de usuario.

✓ Code

```

public static void enviarGrupos(Socket cl) {
    try {
        ObjectOutputStream oos = new
            ↳ ObjectOutputStream(cl.getOutputStream());
        oos.writeObject(grupos);
        oos.flush();

        System.out.println("Grupos enviados");
        oos.close();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
}

```

2.6.4 void recibirHorario(cl)

Ayuda a crear el horario que un alumno ha seleccionado en su procedimiento de inscripción. El servidor recibe el número de boleta del alumno actual, los grupos y las materias que un alumno ha inscrito. Creamos un objeto de tipo Horario y enviamos como parámetro el numero de materias inscritas. Buscamos que el número de boleta coincida con alguna de las boletas en el catalogo,

(si se encuentra, significa que ese número de boleta pertenece a un alumno, es decir, al alumno que se esta registrando).

EL servidor registra los grupos, materias y profesores correspondientes al horario del alumno así como las horas por semana que corresponden a la materia.

Finalmente, a modo de práctica, se generan calificaciones aleatorias con un rango de 5 a 10. Hemos establecido que un alumno solo puede inscribirse una vez, por ello si desea volver a inscribirse no lo hará si ya ha finalizado.

✓ Code

```
public static void recibirHorario(Socket cl, DataInputStream dis) {
    try {
        int boletaActual = dis.readInt();
        String[] g;
        String[] m;

        // Recibir objetos
        ObjectInputStream ois = new ObjectInputStream(cl.getInputStream());
        g = (String[]) ois.readObject();
        m = (String[]) ois.readObject();

        int numMateriasInscritas = m.length;
        int inscripcion = 0;

        Horario horarioInscrito = new Horario(numMateriasInscritas);

        for(int i = 0; i < alumnos.length; i++) {
            if(alumnos[i].getBoleta() == boletaActual)
                inscripcion = i;
        }

        System.out.println("Horario recibido:");

        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < numMateriasInscritas; j++) {
                if(grupos[i].getNombre().equals(g[j])) {
                    System.out.print(grupos[i].getNombre() + " - ");
                    horarioInscrito.setGrupos(grupos[i], j);

                    Materia[] mGrupo = grupos[i].getMaterias();
                    String[] profsInscritos = grupos[i].getProfesores();
                    String[][] horasInscritas = grupos[i].getHoras();

                    for(int k = 0; k < 6; k++) {
```

```
        if(mGrupo[k].getNombre().equals(m[j])) {

            System.out.print(mGrupo[k].getNombre() + " - ");
            horarioInscrito.setMaterias(mGrupo[k], j);
            horarioInscrito.setProfesores(profsInscritos[k], j);
            System.out.print(profsInscritos[k] + " - ");

            String hrs = "";

            for(int dias = 0; dias < 5; dias++) {
                horarioInscrito.setHoras(horasInscritas[k][dias],
                    ↪ j, dias);
                hrs = hrs + " " + horasInscritas[k][dias];
            }
            System.out.println(hrs);
        }
    }

    // Genera calificaciones aleatorias a modo de
    ↪ ejemplo.
    // Rango de 5 a 10
    int calif = (int) (Math.random() * 6) + 5;
    horarioInscrito.setCalifs(calif, j);
}

}

System.out.println("Construyendo horario... Listo.");
System.out.println("Asignando horario a alumno.");

alumnos[inscripcion].setHorario(horarioInscrito);
alumnos[inscripcion].setInscripcion(true);

System.out.println("Alumno inscrito con horario correctamente.");
ois.close();
cl.close();
}
catch(Exception e) {
    e.printStackTrace();
} // Fin catch
}
```

2.7 Cliente

Esta clase es la encargada de solicitar todas las peticiones al servidor, y pasarle las respuestas a la interfaz de usuario correspondiente, que las irá desplegando de forma gráfica al usuario.

Primero se inicializa el puerto del servidor, el host, y un objeto grupo para manejar los grupos que se pueden ver como objetos globales y estáticos.

Toda la funcionalidad importante de las interfaces de usuario Login, Inscripción, ver horario y ver calificaciones, se encuentra en los métodos de esta clase (todos estáticos para ser invocados desde las clases de interfaces, los cuales son:

2.7.1 Alumno iniciarSesion(boletoTmp, psswdTemp)

Cuando el usuario decide iniciar sesión, después de ingresar su número de boleto y contraseña, el cliente solicita al servidor por medio de un socket y enviando una bandera entera de valor 0, la autenticación de usuario.

Para que el servidor pueda llevar a cabo dicha función, el cliente le envía la boleto y contraseña desde los textbox de la interfaz de usuario. Si la sesión inicio con éxito, entonces se recibe un objeto de tipo **Alumno**, que tiene como número de boleto, la boleto que se ingreso.

Dicho alumno es el que la función regresa.

✓Code

```
public static Alumno iniciarSesion(int boleto_tmp, String psswd_temp) {
    Alumno alumnoActual = null;

    try {
        Socket cl = new Socket(host, pto);
        DataOutputStream dos = new
            ↳ DataOutputStream(cl.getOutputStream()); //Enviar
            ↳ datos

        // Bandera con valor 0 = Login
        dos.writeInt(0);
        dos.flush();
        System.out.println("Enviando datos: " + boleto_tmp +
            ↳ " - " + psswd_temp + ", esperando servidor...");

        // Enviar boleto y psswd de los textbox
        dos.writeInt(boleto_tmp);
        dos.flush();
        dos.writeUTF(psswd_temp);
        dos.flush();
    }
```

```

        // Recibir objetos
        ObjectInputStream ois = new
        ↪ ObjectInputStream(cl.getInputStream());

        // Importante hacer un cast
        alumnoActual = (Alumno) ois.readObject();
        System.out.println("Objeto recibido");

        dos.close();
        ois.close();
        cl.close();
    }
    catch(Exception e) {
        e.printStackTrace();
    } // Catch

    return alumnoActual;
}

```

2.7.2 void obtenerGrupos()

Cuando el usuario elige la opción de **Inscripción**, si el alumno aún no ha hecho su inscripción entonces se muestra la interfaz de usuario para inscribirse

Al mostrar la interfaz de usuario de inscripción, el cliente solicita al servidor por medio de un socket y enviando una bandera entera de valor 1, mostrar todos los grupos existentes en el catalogo. El cliente recibe dichos grupos.

✓Code

```

public static void obtenerGrupos() {
    try {
        Socket cl = new Socket(host, pto);
        // Enviar datos
        DataOutputStream dos = new
        ↪ DataOutputStream(cl.getOutputStream());

        // La bandera tiene el valor de 1 = getGrupos
        dos.writeInt(1);
        dos.flush();

        ObjectInputStream ois = new
        ↪ ObjectInputStream(cl.getInputStream());
        ↪ //Recibir objetos
    }
}

```

```

        grupos = (Grupo[]) ois.readObject();
        System.out.println("Grupos recibidos");

        ois.close();
        cl.close();

    }catch(Exception e) {
        e.printStackTrace();
    } //catch
}

```

2.7.3 void enviarHorario(grupos, materia, boleta)

Cuando el usuario elige la opción de **Inscribirse**, (implica que se encuentra en la interfaz de usuario de inscripción) para este momento ya debe haber seleccionado las materias que desea cursar. El cliente envía al servidor por medio de un socket y enviando una bandera entera de valor 2, el número de boleta del alumno, el objeto grupo y materia para que puedan ser registradas como Horario del alumno correspondiente.

✓ Code

```

public static void enviarHorario(String[] grupos, String[] materias, int
    → boleta) {
    try {
        Socket cl = new Socket(host, pto);

        DataOutputStream dos = new
            → DataOutputStream(cl.getOutputStream()); //Enviar
            → datps

        // La bandera tiene el valor de 2 = enviarHorario
        dos.writeInt(2);
        dos.flush();

        // Enviamos el numero de boleta
        dos.writeInt(boleta);
        dos.flush();
        System.out.println("Boleta enviada");
        // Enviamos los arreglos de materias y grupos
        → seleccionados
        ObjectOutputStream oos = new
            → ObjectOutputStream(cl.getOutputStream());
        oos.writeObject(grupos);
        oos.flush();
    }
}

```

```
        System.out.println("Grupos seleccionados enviados");
        oos.writeObject(materias);
        oos.flush();
        System.out.println("Materias seleccionadas
        → enviadas");
        dos.close();
        oos.close();
        cl.close();
    }
    catch(Exception e) {
        e.printStackTrace();
    } //catch
}
```

2.8 Interfaces de Usuario

A continuación se describen las interfaces de usuario usadas para que un alumno pueda iniciar sesión, inscribirse, ver su horario, ver sus calificaciones y cerrar sesión.

Las interfaces de usuario, en general, están construidas con un contenedor, paneles y diversos elementos.

2.8.1 Login

Contiene un panel para mostrar los campos de entrada de número de boleta y contraseña. El alumno deberá ingresar su número de boleta y contraseña para poder tener acceso y evidentemente deben ser datos válidos y correctos.

Al seleccionar el botón de iniciar sesión se verifica que los campos de texto no estén vacíos, cumpliéndose esa condición se obtiene el número de boleta y contraseña.

Se envía al cliente el número de boleta y contraseña introducidos para que el servidor valide que sean correctos y como resultado regresen un objeto de tipo Alumno para ser manejado en las diferentes funciones que se tienen en toda la aplicación.

Si se aprueba la sesión, entonces se muestra la siguiente interfaz de usuario, llamada MiniMenu.

✓ Code

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;

public class Login extends JFrame implements ActionListener {
    JPanel panelBotones, panelEntrada;
```

```
        JTextField TxtBoleta;
        JPasswordField TxtPassword;
        JLabel [] l;
        JButton BtnIniciar;
        String[] label = {"Boleta:", "Contraseña:"};

        public Login() {
            int i;
            Container c = getContentPane();
            c.setLayout(new BoxLayout(c, BoxLayout.Y_AXIS));

            // Para panel de entradas
            panelEntrada = new JPanel();
            panelEntrada.setLayout(new BoxLayout(panelEntrada,
                ↳ BoxLayout.Y_AXIS));
            panelEntrada.setBorder(BorderFactory.createEmptyBorder(10,
                ↳ 10, 0, 10));
            l = new JLabel[2];

            for(i = 0; i < 2; i++) {
                l[i] = new JLabel(label[i]);
            }

            TxtBoleta = new JTextField();
            TxtPassword = new JPasswordField();
            TxtBoleta.setAlignmentX(Component.CENTER_ALIGNMENT);
            TxtPassword.setAlignmentX(Component.CENTER_ALIGNMENT);

            panelEntrada.add(l[0]);
            panelEntrada.add(TxtBoleta);
            panelEntrada.add(l[1]);
            panelEntrada.add(TxtPassword);

            c.add(panelEntrada);

            // Para panel de botones
            panelBotones = new JPanel();
            panelBotones.setLayout(new BoxLayout(panelBotones,
                ↳ BoxLayout.Y_AXIS));
            panelBotones.setBorder(BorderFactory.createEmptyBorder(10,
                ↳ 45, 1000, 10));
            BtnIniciar = new JButton("Iniciar Sesión");
            BtnIniciar.setAlignmentX(Component.CENTER_ALIGNMENT);
```



```
panelBotones.add(BtnIniciar);

c.add(panelBotones);

BtnIniciar.addActionListener(this);

}

public void actionPerformed(ActionEvent e) {
    JButton b = (JButton) e.getSource();

    if(b == BtnIniciar) {
        // Iniciar sesion
        if(TxtBoleta.getText().length() == 0 ||
           TxtPassword.getPassword().length == 0) {
            JOptionPane.showMessageDialog(null, "Ingrese
            sus datos.");
        }
        else {
            char[] pass = TxtPassword.getPassword();
            String passwd_temp = String.valueOf(pass);
            int boleta_temp = 0;

            try {
                boleta_temp =
                    Integer.parseInt(TxtBoleta.getText());
                Alumno alumno =
                    Cliente.iniciarSesion(boleta_temp,
                    passwd_temp);

                if(alumno != null){
                    System.out.println("Objeto
                    alumno recibido con
                    boleta: " +
                    alumno.getBoleta());

                    /*Construir objeto MiniMenu
                    y abrirlo*/
                    crearMiniMenu(alumno);
                    /*Cerrar login*/
                    System.out.print("Cerrando
                    Login...");
                    this.setVisible(false);
                }
            }
        }
    }
}
```

```
        System.out.println("
        ↪ Cerrado.");
        this.dispose();
    }
    else {
        System.out.println("Objeto
        ↪ recibido es null");
        JOptionPane.showMessageDialog(null,
        ↪ "Contrasena y/o boleta
        ↪ incorrectas. Intente de
        ↪ nuevo.", "Datos
        ↪ incorrectos",
        ↪ JOptionPane.ERROR_MESSAGE);
        TxtBoleta.setText("");
        TxtPassword.setText("");
    }
}
catch(Exception ex) {
    JOptionPane.showMessageDialog(null,
    ↪ "El campo de boleta solo admite
    ↪ numeros.", "Datos incorrectos",
    ↪ JOptionPane.ERROR_MESSAGE);
    TxtBoleta.setText("");
    TxtPassword.setText("");
}
}
}

public static void crearMiniMenu(Alumno alumno) {
    MiniMenu menu = new MiniMenu(alumno);
    System.out.println("Enviando objeto alumno a MiniMenu,
    ↪ abriendo MiniMenu....");
    menu.setTitle("MENU");
    menu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    menu.setSize(500, 200);
    menu.setVisible(true);
    menu.setLocationRelativeTo(null);
}

public static void main(String s[]) {
    Login f = new Login();
    f.setTitle("Iniciar sesion");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

```
f.setSize(400, 200);  
f.setVisible(true);  
f.setLocationRelativeTo(null);  
}  
}
```

2.8.2 MiniMenu

Esta interfaz de usuario, es mostrada cuando la autenticación del inicio de sesión fue aprobada. De tal forma que muestra las siguientes opciones:

- Inscribir
- Ver horario
- Ver calificaciones
- Cerrar sesión

Existe un botón para cada una de esas opciones que al ser presionado, genera su funcionalidad, las cuales son descritas a continuación:

- **Inscribir**

Se verifica que el alumno no se haya inscrito, puesto que se ha delimitado a una única instrucción. Si cumple con dicha condición, entonces se muestra la interfaz de usuario inscribir (esta pantalla es descrita en una sección posterior).

- **Ver horario**

Se verifica que el alumno ya se haya inscrito para poder ver su horario, de otro modo no podrá ingresar a esta opción. Si cumple con dicha condición, entonces se muestra la interfaz de usuario para ver Horario (esta pantalla es descrita en una sección posterior).

- **Ver calificaciones**

Se verifica que el alumno ya se haya inscrito para poder ver sus calificaciones, de otro modo no podrá ingresar a esta opción. Si cumple con dicha condición, entonces se muestra la interfaz de usuario para ver Calificaciones (esta pantalla es descrita en una sección posterior). Cabe destacar que las calificaciones son puestas aleatoriamente puesto que no existe el modulo de agregar calificaciones.

- **Cerrar sesión**

Se cierra sesión y regresa a la pantalla del login.

✓ Code

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;

public class MiniMenu extends JFrame implements ActionListener {
    JPanel panelBotones;
    JButton BtnInscribir, BtnCal, BtnHorario, BtnCerrarSesion;
    Alumno alumno = null;

    public MiniMenu(Alumno alumno) {
        //Vamos pasando el objeto alumno, esto a modo de una
        ↪ "sesion"
        this.alumno = alumno;
        Container c = getContentPane();
        c.setLayout(new BoxLayout(c, BoxLayout.Y_AXIS));

        panelBotones = new JPanel();
        panelBotones.setLayout(new BoxLayout(panelBotones,
        ↪ BoxLayout.X_AXIS));
        panelBotones.setBorder(BorderFactory.createEmptyBorder(60,
        ↪ 0, 0, 0));

        BtnInscribir = new JButton("Inscribir");
        BtnCal = new JButton("Ver calificaciones");
        BtnHorario = new JButton("Ver horario");
        BtnCerrarSesion = new JButton("Cerrar sesion");

        panelBotones.add(BtnInscribir);
        panelBotones.add(BtnCal);
        panelBotones.add(BtnHorario);
        panelBotones.add(BtnCerrarSesion);

        c.add(panelBotones);

        BtnInscribir.addActionListener(this);
        BtnCal.addActionListener(this);
        BtnHorario.addActionListener(this);
        BtnCerrarSesion.addActionListener(this);
    }

    public static void crearLogin() {
        Login f = new Login();
        f.setTitle("Iniciar sesion");
    }
}
```

```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setSize(400, 200);
f.setVisible(true);
f.setLocationRelativeTo(null);
}

public static void crearInscribir(Alumno alumno) {
    InscribirV f = new InscribirV(alumno);
    System.out.println("Enviando objeto alumno a inscribir,
        ↪ abriendo inscribir....");
    f.setTitle("Inscribir");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(700, 720);
    f.setVisible(true);
    f.setLocationRelativeTo(null);
}

public static void crearHorario(Alumno alumno) {
    HorarioV f = new HorarioV(alumno);
    System.out.println("Enviando objeto alumno a ver Horario,
        ↪ abriendo Horario....");
    f.setTitle("Horario");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(700, 450);
    f.setVisible(true);
    f.setLocationRelativeTo(null);
}

public static void crearCalificaciones(Alumno alumno) {
    CalificacionesV f = new CalificacionesV(alumno);
    System.out.println("Enviando objeto alumno a ver Horario,
        ↪ abriendo Horario....");
    f.setTitle("Calificaciones");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(700, 450);
    f.setVisible(true);
    f.setLocationRelativeTo(null);
}

public void actionPerformed(ActionEvent e) {
    JButton b = (JButton) e.getSource();

    if(b == BtnInscribir) {
        if(alumno.getInscripcion()){
```

```
// Abrir ventana de Ver horario
JOptionPane.showMessageDialog(null, "Su
↳ inscripcion ha finalizado.",
↳ "Inscripcion finalizada",
↳ JOptionPane.ERROR_MESSAGE);
}
else {
    // Abrir ventana para inscribirse
    crearInscribir(alumno);
    System.out.print("Cerrando MiniMenu....");
    this.setVisible(false);
    System.out.println(" Cerrado.");
    this.dispose();
}
}
else if(b == BtnHorario) {
    if(alumno.getInscripcion()) {
        // Abrir ventana de Ver horario
        crearHorario(alumno);
        System.out.print("Cerrando MiniMenu....");
        this.setVisible(false);
        System.out.println(" Cerrado.");
        this.dispose();
    }
    else {
        JOptionPane.showMessageDialog(null, "Debe
↳ finalizar su inscripcion para ver su
↳ horario.", "Inscripcion no finalizada",
↳ JOptionPane.ERROR_MESSAGE);
    }
}
else if(b == BtnCal) {
    if(alumno.getInscripcion()) {
        // Abrir ventana de Ver horario
        crearCalificaciones(alumno);
        System.out.print("Cerrando MiniMenu....");
        this.setVisible(false);
        System.out.println(" Cerrado.");
        this.dispose();
    }
    else {
```

```

        JOptionPane.showMessageDialog(null, "Debe
        ↪ finalizar su inscripcion para ver su
        ↪ horario.", "Inscripcion no finalizada",
        ↪ JOptionPane.ERROR_MESSAGE);
    }
}
else if(b == BtnCerrarSesion){
    //Cerrar sesion, abrir login
    crearLogin();
    System.out.print("Cerrando MiniMenu...");
    this.setVisible(false);
    System.out.println(" Cerrado.");
    alumno = null;
    this.dispose();
}
}
}
}

```

2.8.3 Inscribir

Esta pantalla se muestra cuando el usuario ha seleccionado la opción de inscribir desde el Mini Menu.

Al crear esta pantalla, se recibe como parámetro un objeto de tipo Alumno, que nos ayuda a obtener todos los datos del alumno que ha iniciado sesión y desea inscribirse.

Hablando técnicamente, se crean 7 diferentes paneles para que aparezcan en la Ventana.

En la sección **Datos personales** se muestra la foto del alumno, su número de boleta y su nombre completo.

A continuación, el alumno puede elegir uno de los grupos que aparecen en el comboBox. El objetivo de esta funcionalidad es pueda visualizar las materias que pertenecen al grupo que selección de tal forma que pueda seleccionar alguna o muchas de ellas para agregarlas a su horario. De manera interna, se envía al cliente el id del grupo para que este le pida al servidor el grupo que estamos solicitando, de tal forma que el cliente recibe el grupo.

La siguiente sección, es la que muestra el horario que ha seleccionado hasta el momento, cada que el agregue una materia, aparecerá ahí. Simulando que esta formando su horario.

Finalmente, existen dos botones:

- **Inscribir:** Permite que se inscriba el horario mostrado hasta el momento, para poder inscribir, se envía al Cliente los grupos, materias y número de boleta del alumno para que los envíe al servidor y este registre el horario que el alumno a elegido.
- **Eliminar:** Elimina las filas seleccionadas del Horario que el alumno esta generando.

Para que se puedan cargar los datos, después de haber confirmado la inscripción, se pide iniciar sesión nuevamente. Después de haber hecho eso, el alumno ya puede ver su horario y sus calificaciones.

✓Code

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.util.*;
import java.lang.reflect.Field;
import javax.swing.table.*;

public class InscribirV extends JFrame implements ActionListener {
    JPanel panelInfo, panelFoto, panelDatos, panelBuscar, panelMostrar,
        ↪ panelHorario, panelBotones;
    JButton btnBuscar, btnAgregar, btnInscribir, btnEliminar;
    JLabel lfoto, lGrupo, lNombre, lBoleta, nombre, boleta;
    JComboBox<String> comboBox;
    JScrollPane scrollMostrar, scrollHorario;
    ImageIcon foto;

    // Guarda la lista de grupos que un estudiante inscribe
    static JList<String> stringGrupo;
    // Guarda la lista de materias de un grupo
    static JList<String> stringMateria;
    static DefaultListModel<String> modelo;
    JTable tablaMostrar, tablaHorario;

    //Modelos para agregar filas a las tablas
    DefaultTableModel modeloMostrar;
    DefaultTableModel modeloHorario;

    Alumno alumno = null;

    public InscribirV(Alumno alumno) {
        //Vamos pasando el objeto alumno, esto a modo de una
        ↪ "sesion"
        this.alumno = alumno;

        Container c = getContentPane();
        c.setLayout(new FlowLayout());
    }
}
```



```

//
↪ -----
//
↪ FOTO
//
↪ -----

// Agregamos al panel DATOS la información del usuario
panelFoto = new JPanel(new GridLayout(1, 1));
panelFoto.setPreferredSize(new Dimension(100, 100));

// Agregamos al panel FOTO la imagen default
lfoto = new JLabel(foto);
lfoto.setIcon(new
↪ ImageIcon(alumno.getFoto().getImage().getScaledInstance(100,
↪ 100, Image.SCALE_SMOOTH)));
//lfoto.setIcon(new ImageIcon(new
↪ javax.swing.ImageIcon(getClass().getResource("fotos/2014171285.png"),
↪ 100, Image.SCALE_SMOOTH)));
panelFoto.add(lfoto);

//
↪ -----
//
↪ DATOS PERSONALES
//
↪ -----

// Agregamos al panel DATOS la información del usuario
panelDatos = new JPanel(new GridLayout(2, 2));
panelDatos.setBorder(BorderFactory.createTitledBorder("DATOS
↪ PERSONALES"));
panelDatos.setPreferredSize(new Dimension(300, 100));

lBoleta = new JLabel("Boleta: ");
boleta = new JLabel("" + alumno.getBoleta());
lNombre = new JLabel("Nombre: ");
nombre = new JLabel(alumno.getNombreCompleto());
panelDatos.add(lBoleta); panelDatos.add(boleta);
panelDatos.add(lNombre); panelDatos.add(nombre);

//
↪ -----

```

```

//                                PANEL QUE INTEGRA FOTO Y
↪  DATOS PERSONALES
//
↪  -----

panelInfo = new JPanel(new GridLayout(1, 2));
panelInfo.setPreferredSize(new Dimension(650, 100));
panelInfo.add(panelFoto); panelInfo.add(panelDatos);

c.add(panelInfo);

//
↪  -----
//                                PANEL
↪  PARA BUSCAR GRUPO
//
↪  -----

// Agregamos al panel GRUPO el comboBox para elegir un grupo
panelBuscar = new JPanel(new GridLayout(1, 3));
panelBuscar.setBorder(BorderFactory.createTitledBorder("SELECCIONA
↪  UN GRUPO"));
panelBuscar.setPreferredSize(new Dimension(650, 50));

lGrupo = new JLabel("GRUPO: ");
comboBox = new JComboBox<>();
btnBuscar = new JButton("Buscar");
btnBuscar.addActionListener(this);
panelBuscar.add(lGrupo); panelBuscar.add(comboBox);
↪  panelBuscar.add(btnBuscar);

c.add(panelBuscar);

//
↪  -----
//                                PANEL PARA MOSTRAR EL
↪  GRUPO QUE SE SELECCIONO
//
↪  -----

panelMostrar = new JPanel(new GridLayout(1, 3));
panelMostrar.setBorder(BorderFactory.createTitledBorder("HORARIO
↪  DEL GRUPO SELECCIONADO"));
panelMostrar.setPreferredSize(new Dimension(650, 200));

```

```

String[] titulos = {"GRUPO", "MATERIA", "PROFESOR", "LUNES",
    ↪ "MARTES", "MIERCOLES", "JUEVES", "VIERNES"};
scrollMostrar = new JScrollPane();
DefaultTableModel modeloMostrar = new
    ↪ DefaultTableModel(null, titulos);
tablaMostrar = new JTable(modeloMostrar);
scrollMostrar.setViewportViewView(tablaMostrar);
Object[] fila = new Object[8];
int i, j;
/*for(i = 0; i < 10; i++) {
    for(j = 0; j < 8; j++) {
        fila[j] = " ";
    }
    modeloMostrar.addRow(fila);
}*/
panelMostrar.add(scrollMostrar);
c.add(panelMostrar);

//
↪ -----
//
↪                                     BOTON DE AGREGAR
↪ SELECCION AL HORARIO
//
↪ -----

btnAgregar = new JButton("Agregar");
btnAgregar.setPreferredSize(new Dimension(100, 35));
btnAgregar.addActionListener(this);
c.add(btnAgregar);

//
↪ -----
//
↪                                     PANEL PARA MOSTRAR EL HORARIO ELEGIDO
↪ HASTA CIERTO MOMENTO
//
↪ -----

panelHorario = new JPanel(new GridLayout(1, 3));
panelHorario.setBorder(BorderFactory.createTitledBorder("HORARIO
    ↪ SELECCIONADO"));
panelHorario.setPreferredSize(new Dimension(650, 200));

scrollHorario = new JScrollPane();

```

```

modeloHorario = new DefaultTableModel(null, titulos);
tablaHorario = new JTable(modeloHorario);
scrollHorario.setViewportViewView(tablaHorario);
fila = new Object[8];
/*for(i = 0; i < 1; i++) {
    for(j = 0; j < 8; j++) {
        fila[j] = " ";
    }
    modeloHorario.addRow(fila);
}*/
panelHorario.add(scrollHorario);
c.add(panelHorario);
c.add(panelHorario);

//
↪ -----
//                                     PANEL PARA BOTON
↪  DE INSCRIBIR Y ELIMINAR
//
↪ -----

panelBotones = new JPanel(new GridLayout(1, 2));
panelBotones.setPreferredSize(new Dimension(650, 40));

btnInscribir = new JButton("Inscribir");
btnInscribir.setPreferredSize(new Dimension(100, 35));
btnInscribir.addActionListener(this);
btnEliminar = new JButton("Eliminar");
btnEliminar.setPreferredSize(new Dimension(100, 35));
btnEliminar.addActionListener(this);

panelBotones.add(btnInscribir);
↪ panelBotones.add(btnEliminar);
c.add(panelBotones);

//
↪ -----
//                                     DESCARGAR GRUPOS
↪  DESDE EL SERVIDOR
//
↪ -----

Cliente.obtenerGrupos();

for(j = 0; j < Cliente.grupos.length; j++){
    String grupo = Cliente.grupos[j].getNombre();

```

```
        comboBox.addItem(grupo);
    }

}

public void crearLogin(){
    Login f = new Login();
    f.setTitle("Iniciar sesion");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setSize(400, 200);
    f.setVisible(true);
    f.setLocationRelativeTo(null);
}

public void actionPerformed(ActionEvent e) {
    JButton b = (JButton) e.getSource();

    if(b == btnBuscar) {
        // Hace que se permita ver el horario de un grupo
        ↪ seleccionado
        int idGrupo = comboBox.getSelectedIndex();
        Grupo g = Cliente.grupos[idGrupo];
        String nombre = g.getNombre();
        Materia[] materias = g.getMaterias();
        String[] profs = g.getProfesores();
        String[] [] horas = g.getHoras();

        DefaultTableModel modelo = (DefaultTableModel)
            ↪ tablaMostrar.getModel();
        modelo.setRowCount(0);

        for(int i = 0; i < materias.length; i++){
            String[] fila = {nombre,
                ↪ materias[i].getNombre(), profs[i],
                ↪ horas[i][0], horas[i][1], horas[i][2],
                ↪ horas[i][3], horas[i][4]};
            modelo.addRow(fila);
        }
        System.out.println("Grupo desplegado actualizado.");
    }
    if(b == btnAgregar) {
        // Se agregan al horario del usuario las materias
        ↪ seleccionado
    }
}
```

```
DefaultTableModel modelo1 = (DefaultTableModel)
    ↳ tablaMostrar.getModel();
int[] seleccion = tablaMostrar.getSelectedRows();
String[] fila = new String[8];
DefaultTableModel modelo2 = (DefaultTableModel)
    ↳ tablaHorario.getModel();

for(int i = 0; i < seleccion.length; i++){
    for(int j = 0; j < 8; j++){
        fila[j] = (String)
            ↳ modelo1.getValueAt(seleccion[i],
            ↳ j);
    }
    modelo2.addRow(fila);
}
System.out.println("Materias agregadas.");
}

if(b == btnInscribir) {
    int resp = JOptionPane.showConfirmDialog(null, "Esta
    ↳ seguro de inscribir el siguiente horario?",
    ↳ "Confirmacion inscripcion",
    ↳ JOptionPane.YES_NO_OPTION);

    if(resp == JOptionPane.YES_OPTION){
        // Se inscriben al horario del usuario las
        ↳ materias seleccionadas
        DefaultTableModel modelo2 =
            ↳ (DefaultTableModel)
            ↳ tablaHorario.getModel();
        int numMaterias =
            ↳ tablaHorario.getRowCount();
        String[] grupos = new String[numMaterias];
        String[] materias = new String[numMaterias];

        for(int i = 0; i < numMaterias; i++){
            grupos[i] = (String)
                ↳ modelo2.getValueAt(i, 0);
            materias[i] = (String)
                ↳ modelo2.getValueAt(i, 1);
        }
    }
}
```

```

        //Enviamos el horario al servidor para que
        ↪ este inscriba al alumno dentro de su
        ↪ catalogo
        System.out.println("Enviando horario...");
        Cliente.enviarHorario(grupos, materias,
        ↪ alumno.getBoleta());
        System.out.println("Inscripcion correcta");
        JOptionPane.showMessageDialog(null,
        ↪ "Inscripcion correcta. Debe iniciar
        ↪ nuevamente sesion para ver su horario y
        ↪ calificaciones.");
        crearLogin();
        System.out.print("Cerrando InscribirV....");
        this.setVisible(false);
        System.out.println(" Cerrado.");
        alumno = null;
        this.dispose();
    }
}
if(b == btnEliminar) {
    // Se eliman del horario del usuario las masterias
    ↪ seleccionadas
    DefaultTableModel modelo2 = (DefaultTableModel)
    ↪ tablaHorario.getModel();
    int[] seleccion = tablaHorario.getSelectedRows();

    for(int i = seleccion.length - 1; i >= 0; i--){
        modelo2.removeRow(seleccion[i]);
    }
    System.out.println("Materias eliminadas.");
}
}
}

```

2.8.4 Ver calificaciones

Esta pantalla se muestra cuando el usuario ha seleccionado la opción de Ver calificaciones desde el Mini Menu.

Al crear esta pantalla, se recibe como parámetro un objeto de tipo Alumno, que nos ayuda a obtener todos los datos del alumno que ha iniciado sesión. Hablando técnicamente, se crean un paneles para que aparezcan en la Ventana.

En la sección **Datos personales** se muestra la foto del alumno, su número de boleta y su nombre completo.

Por otro lado en la sección de Calificaciones, se muestran las materias que el alumno ha inscrito. Para mostrarlo, obtenemos el objeto Horario desde el objeto alumno que se ha recibido al crear la pantalla.

Cabe mencionar, que las calificaciones son solo para fines de muestra, puesto que el módulo de agregar calificaciones no ha sido implementado. Finalmente se muestra un botón para regresar al Mini Menú.

✓Code

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.util.*;
import java.lang.reflect.Field;
import javax.swing.table.*;

public class CalificacionesV extends JFrame implements ActionListener {
    JPanel panelInfo, panelFoto, panelDatos, panelHorario, panelMostrar;
    JButton btnRegresar;
    JLabel lfoto, lGrupo, lNombre, lBoleta, nombre, boleta;
    JScrollPane scrollHorario, scrollMostrar;
    ImageIcon foto;

    static DefaultListModel<String> modelo;
    JTable tablaMostrar, tablaHorario;

    // Modelos para agregar filas a las tablas
    DefaultTableModel modeloMostrar;
    DefaultTableModel modeloHorario;

    Alumno alumno = null;

    public CalificacionesV(Alumno alumno) {
        //Vamos pasando el objeto alumno, esto a modo de una
        ↪ "sesion"
        this.alumno = alumno;

        Container c = getContentPane();
        c.setLayout(new FlowLayout());

        //
        ↪ -----
```



```

// PANEL DE
↪ FOTO
//
↪ -----

// Agregamos al panel DATOS la información del usuario
panelFoto = new JPanel(new GridLayout(1, 1));
panelFoto.setPreferredSize(new Dimension(100, 100));

// Agregamos al panel FOTO la imagen default
lfoto = new JLabel(foto);
lfoto.setIcon(new
↪ ImageIcon(alumno.getFoto().getImage().getScaledInstance(100,
↪ 100, Image.SCALE_SMOOTH)));
panelFoto.add(lfoto);

//
↪ -----
// PANEL DE
↪ DATOS PERSONALES
//
↪ -----

// Agregamos al panel DATOS la información del usuario
panelDatos = new JPanel(new GridLayout(2, 2));
panelDatos.setBorder(BorderFactory.createTitledBorder("DATOS
↪ PERSONALES"));
panelDatos.setPreferredSize(new Dimension(300, 100));

lBoleta = new JLabel("Boleta: ");
boleta = new JLabel("" + alumno.getBoleta());
lNombre = new JLabel("Nombre: ");
nombre = new JLabel(alumno.getNombreCompleto());
panelDatos.add(lBoleta); panelDatos.add(boleta);
panelDatos.add(lNombre); panelDatos.add(nombre);

//
↪ -----
// PANEL QUE INTEGRA FOTO Y
↪ DATOS PERSONALES
//
↪ -----

panelInfo = new JPanel(new GridLayout(1, 2));

```

```

panelInfo.setPreferredSize(new Dimension(650, 100));
panelInfo.add(panelFoto); panelInfo.add(panelDatos);

c.add(panelInfo);

//
↪ -----
//                                     PANEL PARA MOSTRAR EL
↪ GRUPO QUE SE SELECCIONO
//
↪ -----

panelMostrar = new JPanel(new GridLayout(1, 3));
panelMostrar.setBorder(BorderFactory.createTitledBorder("HORARIO
↪ DEL GRUPO SELECCIONADO"));
panelMostrar.setPreferredSize(new Dimension(650, 200));

String[] titulos = {"GRUPO", "MATERIA", "PROFESOR",
↪ "CALIFICACION"};
scrollMostrar = new JScrollPane();
DefaultTableModel modeloMostrar = new
↪ DefaultTableModel(null, titulos);
tablaMostrar = new JTable(modeloMostrar);
scrollMostrar.setViewportViewView(tablaMostrar);
int i, j;

// ***** HACER QUE SE VEA EL HORARIO
↪ *****

// Hace que se permita ver el horario de un grupo
↪ seleccionado
Horario h = alumno.getHorario();
Grupo[] grupos = h.getGrupos();
Materia[] materias = h.getMaterias();
String[] profs = h.getProfesores();
int[] califs = h.getCalifs();

DefaultTableModel modelo = (DefaultTableModel)
↪ tablaMostrar.getModel();
modelo.setRowCount(0);

for(i = 0; i < materias.length; i++) {

```

```

        System.out.println("Despliego materia del grupo: " +
            ↳ grupos[i].getId());
        Grupo g = Cliente.grupos[grupos[i].getId()];
        String nombreGrupo = g.getNombre();
        String[] filaA = {nombreGrupo,
            ↳ materias[i].getNombre(), profs[i], " "
            ↳ +califs[i]};
        modelo.addRow(filaA);
    }
    System.out.println("Grupo desplegado actualizado.");

    panelMostrar.add(scrollMostrar);
    c.add(panelMostrar);

    //
    ↳ -----
    //                                     BOTON DE AGREGAR
    ↳ SELECCION AL HORARIO
    //
    ↳ -----

    btnRegresar = new JButton("Regresar");
    btnRegresar.setPreferredSize(new Dimension(100, 35));
    btnRegresar.addActionListener(this);
    c.add(btnRegresar);

}

public void actionPerformed(ActionEvent e) {
    JButton b = (JButton) e.getSource();

    if(b == btnRegresar) {
        // Construir objeto MiniMenu y abrirlo
        crearMiniMenu(alumno);
        // Cerrar login
        System.out.print("Cerrando VerHorario....");
        this.setVisible(false);
        System.out.println(" Cerrado.");
        this.dispose();
    }

}

// Crear ventana de MiniMenu
public static void crearMiniMenu(Alumno alumno) {
    MiniMenu menu = new MiniMenu(alumno);

```

```

        System.out.println("Enviando objeto alumno a MiniMenu,
        ↪ abriendo MiniMenu....");
        menu.setTitle("MENU");
        menu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        menu.setSize(500, 200);
        menu.setVisible(true);
        menu.setLocationRelativeTo(null);
    }
}

```

2.8.5 Ver horario

Esta pantalla se muestra cuando el usuario ha seleccionado la opción de Ver calificaciones desde el Mini Menu.

Al crear esta pantalla, se recibe como parámetro un objeto de tipo Alumno, que nos ayuda a obtener todos los datos del alumno que ha iniciado sesión. Hablando técnicamente, se crean un paneles para que aparezcan en la Ventana.

En la sección **Datos personales** se muestra la foto del alumno, su número de boleta y su nombre completo. Por otro lado en la sección de Horario, se muestra el horario que el alumno ha inscrito. Para mostrarlo, obtenemos el objeto horario desde el objeto alumno que se ha recibido al crear la pantalla.

Finalmente se muestra un botón para regresar al Mini Menú.

✓ Code

```

import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.Image;
import javax.swing.ImageIcon;
import java.util.*;
import java.lang.reflect.Field;
import javax.swing.table.*;
public class HorarioV extends JFrame implements ActionListener {
    JPanel panelInfo, panelFoto, panelDatos, panelHorario, panelMostrar;
    JButton btnRegresar;
    JLabel lfoto, lGrupo, lNombre, lBoleta, nombre, boleta;
    JScrollPane scrollHorario, scrollMostrar;
    ImageIcon foto;

    // Guarda la lista de grupos que un estudiante inscribe
    static JList<String> stringGrupo;

```

```

// Guarda la lista de materias de un grupo
static JList<String> stringMateria;
static DefaultListModel<String> modelo;
JTable tablaMostrar, tablaHorario;

// Modelos para agregar filas a las tablas
DefaultTableModel modeloMostrar;
DefaultTableModel modeloHorario;

Alumno alumno = null;

public HorarioV(Alumno alumno) {
    //Vamos pasando el objeto alumno, esto a modo de una
    ↪ "sesion"
    this.alumno = alumno;

    Container c = getContentPane();
    c.setLayout(new FlowLayout());

    //
    ↪ -----
    //
    ↪ PANEL DE
    ↪ FOTO
    //
    ↪ -----

    // Agregamos al panel DATOS la información del usuario
    panelFoto = new JPanel(new GridLayout(1, 1));
    panelFoto.setPreferredSize(new Dimension(100, 100));

    // Agregamos al panel FOTO la imagen default
    lfoto = new JLabel(foto);
    lfoto.setIcon(new
        ↪ ImageIcon(alumno.getFoto().getImage().getScaledInstance(100,
        ↪ 100, Image.SCALE_SMOOTH)));
    panelFoto.add(lfoto);

    //
    ↪ -----
    //
    ↪ PANEL DE
    ↪ DATOS PERSONALES
    //
    ↪ -----

```

```

// Agregamos al panel DATOS la información del usuario
panelDatos = new JPanel(new GridLayout(2, 2));
panelDatos.setBorder(BorderFactory.createTitledBorder("DATOS
↳ PERSONALES"));
panelDatos.setPreferredSize(new Dimension(300, 100));

lBoleta = new JLabel("Boleta: ");
boleta = new JLabel("" + alumno.getBoleta());
lNombre = new JLabel("Nombre: ");
nombre = new JLabel(alumno.getNombreCompleto());
panelDatos.add(lBoleta); panelDatos.add(boleta);
panelDatos.add(lNombre); panelDatos.add(nombre);

//
↳ -----
//                                     PANEL QUE INTEGRA FOTO Y
↳ DATOS PERSONALES
//
↳ -----

panelInfo = new JPanel(new GridLayout(1, 2));
panelInfo.setPreferredSize(new Dimension(650, 100));
panelInfo.add(panelFoto); panelInfo.add(panelDatos);

c.add(panelInfo);

//
↳ -----
//                                     PANEL PARA MOSTRAR EL
↳ GRUPO QUE SE SELECCIONO
//
↳ -----

panelMostrar = new JPanel(new GridLayout(1, 3));
panelMostrar.setBorder(BorderFactory.createTitledBorder("HORARIO
↳ DEL GRUPO SELECCIONADO"));
panelMostrar.setPreferredSize(new Dimension(650, 200));

String[] titulos = {"GRUPO", "MATERIA", "PROFESOR", "LUNES",
↳ "MARTES", "MIERCOLES", "JUEVES", "VIERNES"};
scrollMostrar = new JScrollPane();
DefaultTableModel modeloMostrar = new
↳ DefaultTableModel(null, titulos);
tablaMostrar = new JTable(modeloMostrar);

```

```

scrollMostrar.setViewportViewView(tablaMostrar);
Object[] fila = new Object[8];
int i, j;
/*for(i = 0; i < 10; i++) {
    for(j = 0; j < 8; j++) {
        fila[j] = " ";
    }
    modeloMostrar.addRow(fila);
}*/

// ***** HACER QUE SE VEA EL HORARIO
↪ *****

// Hace que se permita ver el horario de un grupo
↪ seleccionado
Horario h = alumno.getHorario();
int numMaterias = h.getNumMaterias();
Materia[] materias = h.getMaterias();
Grupo[] grupos = h.getGrupos();
String[] profs = h.getProfesores();
String[][] horas = h.getHoras();

DefaultTableModel modelo = (DefaultTableModel)
    ↪ tablaMostrar.getModel();
modelo.setRowCount(0);

for(i = 0; i < materias.length; i++) {
    System.out.println("Despliego materia del grupo: " +
        ↪ grupos[i].getId());
    Grupo g = Cliente.grupos[grupos[i].getId()];
    String nombreGrupo = g.getNombre();
    String[] filaA = {nombreGrupo,
        ↪ materias[i].getNombre(), profs[i], horas[i][0],
        ↪ horas[i][1], horas[i][2], horas[i][3],
        ↪ horas[i][4]};
    modelo.addRow(filaA);
}
System.out.println("Grupo desplegado actualizado.");

panelMostrar.add(scrollMostrar);
c.add(panelMostrar);

```

```

//
↪ -----
//                                     BOTON DE AGREGAR
↪ SELECCION AL HORARIO
//
↪ -----

btnRegresar = new JButton("Regresar");
btnRegresar.setPreferredSize(new Dimension(100, 35));
btnRegresar.addActionListener(this);
c.add(btnRegresar);

}

public void actionPerformed(ActionEvent e) {
    JButton b = (JButton) e.getSource();

    if(b == btnRegresar) {
        /*Construir objeto MiniMenu y abrirlo*/
        crearMiniMenu(alumno);
        /*Cerrar login*/
        System.out.print("Cerrando VerHorario....");
        this.setVisible(false);
        System.out.println(" Cerrado.");
        this.dispose();
    }

}

public static void crearMiniMenu(Alumno alumno) {
    MiniMenu menu = new MiniMenu(alumno);
    System.out.println("Enviando objeto alumno a MiniMenu,
↪ abriendo MiniMenu....");
    menu.setTitle("MENU");
    menu.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    menu.setSize(500, 200);
    menu.setVisible(true);
    menu.setLocationRelativeTo(null);
}

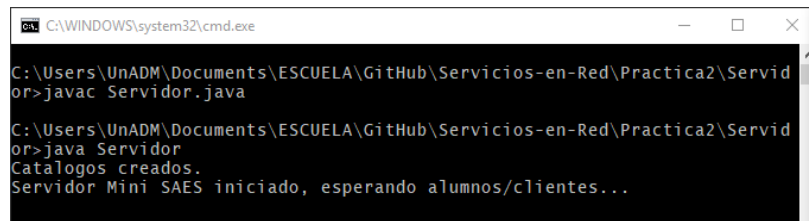
}

```

3 Pruebas

A continuación se muestra el procedimiento para navegar con la aplicación.

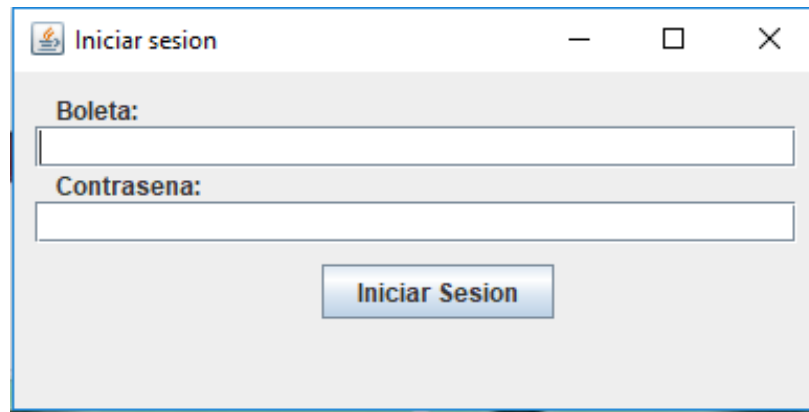
3.1 Servidor



```
C:\WINDOWS\system32\cmd.exe
C:\Users\UnADM\Documents\ESCUELA\GitHub\Servicios-en-Red\Practica2\Servidor>javac Servidor.java
C:\Users\UnADM\Documents\ESCUELA\GitHub\Servicios-en-Red\Practica2\Servidor>java Servidor
Catalogos creados.
Servidor Mini SAES iniciado, esperando alumnos/clientes...
```

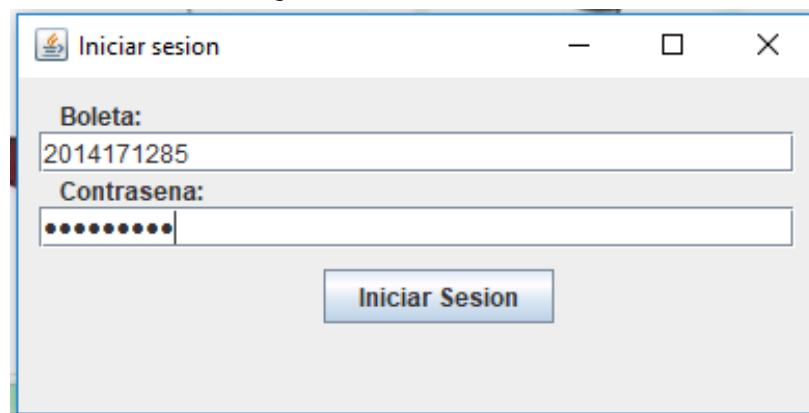
Figure 1: Servidor

3.2 Inicio de sesión



The screenshot shows a window titled "Iniciar sesion" with two input fields. The first field is labeled "Boleta:" and the second is labeled "Contraseña:". Below the fields is a button labeled "Iniciar Sesion".

Figure 2: Inicio de sesión



The screenshot shows the same "Iniciar sesion" window, but now with example data entered. The "Boleta:" field contains the number "2014171285". The "Contraseña:" field contains a series of dots, representing a masked password. The "Iniciar Sesion" button remains at the bottom.

Figure 3: Ejemplo de datos

3.3 Mini menú

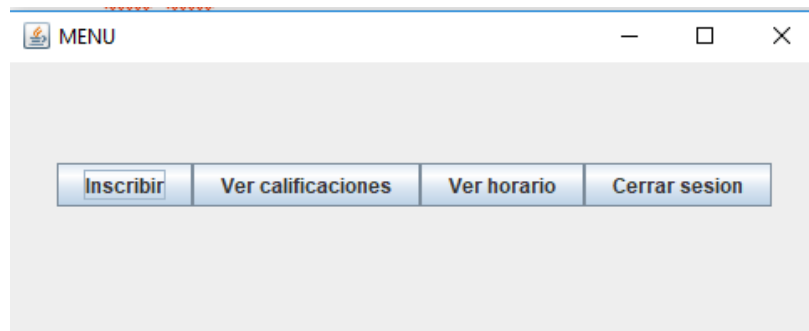


Figure 4: Menú principal

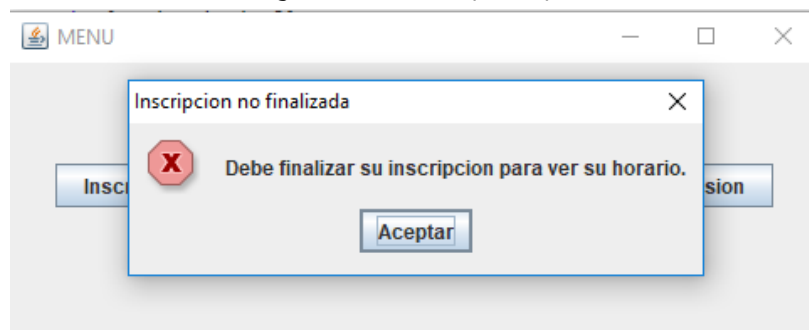


Figure 5: Valida que no se ha inscrito

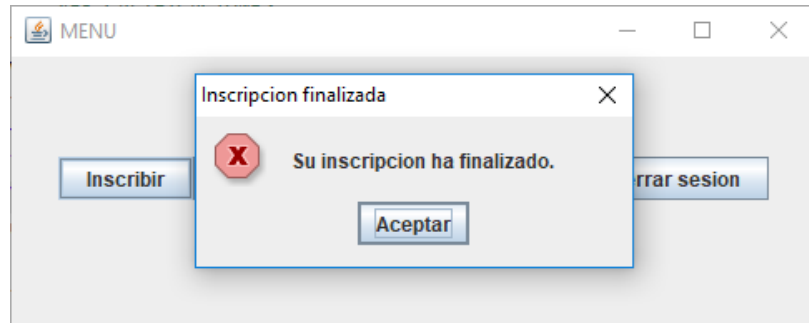




Figure 6: Valida que ya se inscribio

3.4 Inscribir Horario

 Inscribir



DATOS PERSONALES
Boleta: 2014171285
Nombre: Abigail Nicolas Sayago

SELECCIONA UN GRUPO
GRUPO: 3CM8


HORARIO DEL GRUPO SELECCIONADO


GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
-------	---------	----------	-------	--------	-----------	--------	---------

HORARIO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
-------	---------	----------	-------	--------	-----------	--------	---------

Figure 7: Pantalla principal

 Inscribir
 — □ ×



DATOS PERSONALES
 Boleta: 2014171285
 Nombre: Abigail Nicolas Sayago

SELECCIONA UN GRUPO
 GRUPO:


HORARIO DEL GRUPO SELECCIONADO


GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM9	Instrumentaci...	Juan Carlos ...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CM9	Ingenieria Sof...	Jose Jaime ...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM9	Matematicas ...	Ignacio Rios...	8:30 - 10:00		8:30 - 10:00	8:30 - 10:00	
3CM9	Sistemas Op...	Jorge Corte...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM9	Tecnologias p...	Jose Antoni...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30
3CM9	Analisis de Al...	Edgardo Adr...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00

HORARIO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES

Figure 8: Mostrar grupo

 Inscribir
 — □ ×



DATOS PERSONALES
 Boleta: 2014171285
 Nombre: Abigail Nicolas Sayago

SELECCIONA UN GRUPO
 GRUPO:


HORARIO DEL GRUPO SELECCIONADO

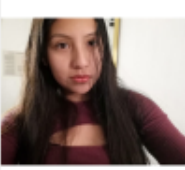
GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM8	Quimica VI	Copca Rami...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CM8	Web Applicati...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Instrumentaci...	Juan Carlos ...	8:30 - 10:00		8:30 - 10:00	8:30 - 10:00	
3CM8	Ingenieria Sof...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Matematicas ...	Ignacio Rios...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30
3CM8	Sistemas Op...	Jorge Corte...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00

HORARIO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM9	Instrumenta...	Juan Carlos...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CV8	Tecnologias...	Jose Antoni...	15:00 - 16:30			15:00 - 16:30	16:30 - 18:00
3CM9	Analisis de ...	Edgardo Adr...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00
3CM8	Ingenieria S...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Web Applica...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Matematica...	Ignacio Rio...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30

Figure 9: Mostrar horario

 Inscribir



DATOS PERSONALES
 Boleta: 2014171285
 Nombre: Abigail Nicolas Sayago


SELECCIONA UN GRUPO

GRUPO: 3CM8 ▼ Buscar

HORARIO DEL GRUPO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM8	Quimica VI	Copca Rami...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CM8	Web Applicati...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Instrumentaci...	Juan Carlos ...	8:30 - 10:00		8:30 - 10:00	8:30 - 10:00	
3CM8	Ingenieria Sof...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Matematicas ...					12:00 - 13:30	12:00 - 13:30
3CM8	Sistemas Op...						10:30 - 12:00

Confirmacion inscripcion ✕

 Esta seguro de inscribir el siguiente horario?

Si No

Agregar

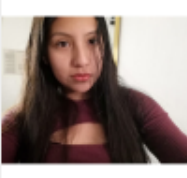
HORARIO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM9	Instrumenta...	Juan Carlos...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CV8	Tecnologias...	Jose Antoni...	15:00 - 16:30			15:00 - 16:30	16:30 - 18:00
3CM9	Analisis de ...	Edgardo Adr...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00
3CM8	Ingenieria S...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Web Applica...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Matematica...	Ignacio Rio...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30

Inscribir
Eliminar

Figure 10: Confirmar

Inscribir



DATOS PERSONALES
 Boleta: 2014171285
 Nombre: Abigail Nicolas Sayago

SELECCIONA UN GRUPO

GRUPO: 3CM8 Buscar

HORARIO DEL GRUPO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM8	Quimica VI	Copca Rami...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CM8	Web Applicati...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Instrumentaci...	Juan Carlos ...	8:30 - 10:00		8:30 - 10:00	8:30 - 10:00	
3CM8	Ingenieria Sof...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	

Mensaje

i Inscripcion correcta. Debe iniciar nuevamente sesion para ver su horario y calificaciones.

Aceptar

Agregar

HORARIO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM9	Instrumenta...	Juan Carlos...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CV8	Tecnologias...	Jose Antoni...	15:00 - 16:30			15:00 - 16:30	16:30 - 18:00
3CM9	Analisis de ...	Edgardo Adr...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00
3CM8	Ingenieria S...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Web Applica...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Matematica...	Ignacio Rio...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30

Inscribir
Eliminar

Figure 11: Aviso

3.5 Ver calificaciones



DATOS PERSONALES
Boleta: 2014171285
Nombre: Abigail Nicolas Sayago

HORARIO DEL GRUPO SELECCIONADO

GRUPO	MATERIA	PROFESOR	CALIFICACION
3CM9	Instrumentacion	Juan Carlos Tellez Barrera	8
3CV8	Tecnologias para la Web	Jose Antonio Ramirez	9
3CM9	Analisis de Algoritmos	Edgardo Adrian Franco Ma...	6
3CM8	Ingenieria Software	Jose Jaime Lopez Rabadan	10
3CM8	Web Application Developm...	Hermes Francisco Montes ...	6
3CM8	Matematicas Avanzadas p...	Ignacio Rios de la Torre	5

Regresar

Figure 12: Calificaciones de un alumno

3.6 Ver horario



DATOS PERSONALES
Boleta: 2014171285
Nombre: Abigail Nicolas Sayago

HORARIO DEL GRUPO SELECCIONADO

GRUPO	MATERIA	PROFESOR	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
3CM9	Instrumentacion	Juan Carlos...	7:00 - 8:30			7:00 - 8:30	8:30 - 10:00
3CV8	Tecnologias par...	Jose Antoni...	15:00 - 16:30			15:00 - 16:30	16:30 - 18:00
3CM9	Analisis de Algo...	Edgardo Adr...		10:30 - 12:00	10:30 - 12:00		10:30 - 12:00
3CM8	Ingenieria Softw...	Jose Jaime ...	10:30 - 12:00	8:30 - 10:00		10:30 - 12:00	
3CM8	Web Application ...	Hermes Fra...		7:00 - 8:30	7:00 - 8:30		7:00 - 8:30
3CM8	Matematicas Av...	Ignacio Rio...	12:00 - 13:30		12:00 - 13:30	12:00 - 13:30	12:00 - 13:30

Regresar

Figure 13: Horario de un alumno

4 Posibles mejoras

1. Crear una base de datos para guardar toda la información.
2. Validar que no inscriba materias repetidas.
3. Validar que no inscriba horas traslapadas.
4. Crear usuarios de tipo profesor.
5. Agregar funcionalidad de ingresar calificaciones.
6. Agregar historial de calificaciones.

5 Conclusiones

5.1 Nicolás Sayago Abigail

Al finalizar esta práctica pude comprender de mejor manera los sockets de flujo puesto que la práctica pasada ya los habíamos usado. Los métodos que se usaron fueron intuitivos y fáciles de utilizar.

Considero que una de las grandes ventajas que como equipo tuvimos, fue que primero discutimos el diseño de la aplicación en general, como las clases, la forma de cargar los datos, los diseños de las interfaces y la asignación de tareas para cada integrante.

En general, manejar los sockets con sus respectivos métodos fue fácil gracias a las clases implementadas de tal forma que al comunicar el servidor con el cliente, se envía y recibían objetos que fueron implementados anteriormente. Uno de los retos a los que nos enfrentamos fue a un diseño agradable e intuitivo para el usuario.

5.2 Ramos Diaz Enrique

El trabajo de comunicación y transmisión de archivos no fue problema, pues la clases Socket y ServerSockets (sockets de flujo bloqueante) en Java hacen todo éste proceso con métodos muy simples de utilizar y comprender.

El verdadero reto fue el manejo de directorios y subdirectorios, tanto para subir como para descargar archivos, ya debíamos tener un control del directorio de trabajo actual, y de distinguirs entre archivos y carpetas cuando el usuario las seleccionaba en el cliente. Hablando del cliente, también hubo algo de complejidad en emular un explorador de archivos como el de Windows o Google Drive al abrir carpetas, ver sus contenidos y navegar por ellas (siendo la única excepción regresar a la ruta anterior).

Nos apoyamos de métodos como recursividad, condicionales, y otros métodos de clases como el de compresión de archivos a descargar en un ZIP, eligiendo esta forma para manejar los archivos y directorios que se descargan desde el servidor.