
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


Alcance


El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos

- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Examen Practico Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar expresiones regulares Entender la cada uno de las características nuevas en Java			
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de Java	
		2. Establecer las características de Java en programación genérica	
		3. Implementar y diseñar los nuevos componentes de programación genérica	
		4. Realizar el informe respectivo según los datos solicitados.	
ACTIVIDADES POR DESARROLLAR (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)			
1. Revisar la teoría y conceptos de Java 8, 9 ,10, 11, 12			
2. Diseñar e implementar las características de Java para generar una expresion regular.			
3. Probar su funcionamiento y rendimiento dentro de los equipos de cómputo de programación genérica.			
4. Realizar práctica codificando los codigos de las nuevas características de Java y su uso dentro de un sistema escolar.			
Enunciado Se desea generar un sistema que me permita extraer infomación del internet a traves de expresiones regulares, esta informacion permitira vincular actividades desarrolladas del los niños con aplicaciones mobiles que permitan apoyar en el desarrollo de las actividades planteadas (https://play.google.com/store?hl=es&gl=US). Adicionalmente, se debe realizar un sistema de gestion de alumnos y actividades planificadas por curso, dentro de este sistema se debe realizar un procesos de administracion de usuarios los mismo que son los docentes de cada curso escolar, en este sentido solo debemos tener un administrador (Rector) el encargado de crear docentes y el curso que se le asigna. Ejemplo Rector: Docentes: 1. Diego Quisi 2. Vladimir Robles 3. Etc. Cursos: 1 de basica 2 de basica 3 basica			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Asignacion de Curso – Docente

1 Basica -> Diego Quisi

2 Basica -> Vladimir Robles

Dentro de cada curso el docente gestionara los estudiantes y las actividades planificadas para el curso, estas actividades tendra una opcion de buscar aplicaciones moviles dentro de la tiendas de play store, obtenidas desde el internet, dentro de esta información lo importante es mostrar el link y una descripción para ello deberán utilizar expresiones regulares.

Ejemplo Docentes:

Alumnos

1. Juan Perez

2. Maria Peralta

3. .

Actividades:

1. Suma de numeros -> Obtener aplicaciones moviles (Link y Titulo)
2. Resta de numeros -> Obtener aplicaciones moviles
3. Oraciones compuestas -> Obtener aplicaciones moviles
4. Etc.

Toda esta infomación sera almacenada dentro de archivos y deberan tener aplicado al menos una patron de diseño y las nuevas características de programación de Java 8 o superior.

Al finalizar, generar el informe de la practica en formato PDF y subir todo el proyecto incluido el informe al repositorio personal.

La fecha de entrega: 23:55 del 01 de diciembre del 2020.

RESULTADO(S) OBTENIDO(S):

Realizar procesos de investigación sobre los cambios importantes de Java

Entender las aplicaciones de codificación de las nuevas características en base a la programación genérica y expresiones regulares.

Entender las funcionalidades adicionales de Java.

CONCLUSIONES:


Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.


RECOMENDACIONES:


Realizar el trabajo dentro del tiempo establecido.

Docente / Técnico Docente: _____

Firma: _____

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA:		ASIGNATURA:	
NRO. PRÁCTICA:		TÍTULO PRÁCTICA:	
OBJETIVO ALCANZADO: Desarrollar un sistema que gestione un sistema de actividades por docente basado en url con un patrón de búsqueda con ayuda de expresiones regulares			
ACTIVIDADES DESARROLLADAS Patrón de diseño Iterador En esta app se utilizo el patrón de diseño Iterador, para obtener una lista de solo el objeto solicitado y cargar los datos a la Tabla de cada una de las ventanas			
<pre> public List<Actividad> actividades() { List<Actividad> lista = new ArrayList(); Actividad actividad; Iterator i = super.getList().iterator(); while (i.hasNext()) { actividad = (Actividad) i.next(); lista.add(actividad); } return lista; } </pre>			
Como se puede mostrar en la foto en ese caso retorna una lista de tipo Actividad			
Para la realización del proyecto se comenzó con los principios de arquitectura de software, el cual consta de un paquete modelo donde están todas las clases relacionadas con programación orientada a objetos, el paquete vista donde se encuentra el diseño de las ventanas JFrame y el paquete controlador, con ayuda de programación genérica maneja los métodos crud para la interacción con los objetos.			
Modelo			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Clase Actividad

```
package ec.edu.ups.modelo;

import java.io.Serializable;

/**
 *
 * @author user
 */
public class Actividad implements Serializable {

    private int codigo;

    private String titulo;

    private String descripcion;

    private String link;

    private Curso curso;

    public Actividad(int codigo, String titulo, String descripcion, String link,
Curso curso) {
        this.codigo = codigo;
        this.titulo = titulo;
        this.descripcion = descripcion;
        this.link = link;
        this.curso = curso;
    }

    public Curso getCurso() {
        return curso;
    }

    public void setCurso(Curso curso) {
        this.curso = curso;
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getTitulo() {
        return titulo;
    }
}
```

```
public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}


public String getLink() {
    return link;
}

public void setLink(String link) {
    this.link = link;
}

@Override
public int hashCode() {
    int hash = 5;
    hash = 97 * hash + this.codigo;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Actividad other = (Actividad) obj;
    if (this.codigo != other.codigo) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Actividad{codigo=").append(codigo);
    sb.append(", titulo=").append(titulo);
    sb.append(", descripcion=").append(descripcion);
    sb.append(", link=").append(link);
    sb.append('}');
    return sb.toString();
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
}
```

```
}
```

Clase Alumno

```
package ec.edu.ups.modelo;

import java.io.Serializable;

/**
 *
 * @author user
 */
public class Actividad implements Serializable {

    private int codigo;

    private String titulo;

    private String descripcion;

    private String link;

    private Curso curso;

    public Actividad(int codigo, String titulo, String descripcion, String link,
Curso curso) {
        this.codigo = codigo;
        this.titulo = titulo;
        this.descripcion = descripcion;
        this.link = link;
        this.curso = curso;
    }

    public Curso getCurso() {
        return curso;
    }

    public void setCurso(Curso curso) {
        this.curso = curso;
    }

    public int getCodigo() {
        return codigo;
    }

    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getTitulo() {
        return titulo;
    }
}
```



```
public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}


public String getLink() {
    return link;
}

public void setLink(String link) {
    this.link = link;
}

@Override
public int hashCode() {
    int hash = 5;
    hash = 97 * hash + this.codigo;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Actividad other = (Actividad) obj;
    if (this.codigo != other.codigo) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Actividad{codigo=").append(codigo);
    sb.append(", titulo=").append(titulo);
    sb.append(", descripcion=").append(descripcion);
    sb.append(", link=").append(link);
    sb.append('}');
    return sb.toString();
}
```

 UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```
}
```

```
}
```

Clase Curso

```
package ec.edu.ups.modelo;

import java.io.Serializable;

/**
 *
 * @author user
 */
public class Curso implements Serializable {

    private int codigo;

    private String nombre;

    private String seccion;
    public Curso(int codigo, String nombre, String seccion) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.seccion = seccion;
    }

    public Curso() {

    }

    public int getCodigo() {
        return codigo;
    }


    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getSeccion() {
        return seccion;
    }

    public void setSeccion(String seccion) {
        this.seccion = seccion;
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

}

@Override
public int hashCode() {
    int hash = 5;
    hash = 97 * hash + this.codigo;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Curso other = (Curso) obj;
    return this.codigo == other.codigo;
}

@Override
public String toString() {
    return "Curso{" + "codigo=" + codigo + ", nombre=" + nombre + ", seccion=" +
seccion + '}';
}
}

```

Clase Docente

```

package ec.edu.ups.modelo;

import java.io.Serializable;

/**
 *
 * @author user
 */
public class Docente extends Persona {


    private String titulo;

    private String tipTitulo;

    private Curso curso;

    private Usuario usuario;

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

public Docente(String cedula, String nombre, String apellido, int edad, String
direccion) {
    super(cedula, nombre, apellido, edad, direccion);
}

public Docente(String cedula, String nombre, String apellido, int edad, String
direccion, String titulo, String tipTitulo) {
    super(cedula, nombre, apellido, edad, direccion);
    this.titulo = titulo;
    this.tipTitulo = tipTitulo;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getTipTitulo() {
    return tipTitulo;
}

public void setTipTitulo(String tipTitulo) {
    this.tipTitulo = tipTitulo;
}

public Curso getCurso() {
    return curso;
}


public void setCurso(Curso curso) {
    this.curso = curso;
}

public Usuario getUsuario() {
    return usuario;
}

public void setUsuario(Usuario usuario) {
    this.usuario = usuario;
}

@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Docente{titulo=").append(titulo);
    sb.append(", tipTitulo=").append(tipTitulo);
    sb.append(", curso=").append(curso);
    sb.append('}');
    return sb.toString();
}

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

}

Clase Usuario

```
public class Usuario implements Serializable {

    private Docente docente;

    private String rol;

    private String correo;

    private String pass;

    public Usuario(Docente docente, String rol, String correo, String pass) {
        this.docente = docente;
        this.rol = rol;
        this.correo = correo;
        this.pass = pass;
    }

    public Usuario() {

    }

    public String getRol() {
        return rol;
    }

    public void setRol(String rol) {
        this.rol = rol;
    }

    public Docente getDocente() {
        return docente;
    }

    public void setDocente(Docente docente) {
        this.docente = docente;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }

    public String getPass() {
        return pass;
    }

    public void setPass(String pass) {
```

```
        this.pass = pass;
    }

    @Override
    public String toString() {
        return "Usuario{" + "docente=" + docente + ", rol=" + rol + ", correo=" +
correo + ", pass=" + pass + '}';
    }
}
```

Paquete controlador

Clase AbstractControlador

```
package ec.edu.ups.controlador;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

/**
 *
 * @author user
 * @param <E>
 */
public abstract class AbstractControlador<E> {

    private String ruta;
    private List<E> lista;

    public AbstractControlador(String ruta) {
        lista = new ArrayList();
        this.ruta = ruta;
        //cargarDatos();
    }

    public void cargarDatos() throws ClassNotFoundException, IOException {
        ObjectInputStream datos = null;
        try {
            File f = new File(ruta);
            FileInputStream a = new FileInputStream(f);
            datos = new ObjectInputStream(a);

            lista = (List<E>) datos.readObject();

        } catch (IOException e) {
```

```
}

}

public void guardarDatos(String ruta) throws IOException {
    ObjectOutputStream datos = null;
    File f = new File(ruta);
    FileOutputStream archivo = new FileOutputStream(f);
    datos = new ObjectOutputStream(archivo);
    datos.writeObject(lista);
}

public boolean crear(E objeto) {

    if (validar(objeto) == true) {
        return lista.add(objeto);
    }
    return false;
}

public Optional<E> buscar(E comparar) {
    return lista.stream().filter(objeto -> objeto.equals(comparar)).findFirst();
}

public int posicion(E objetoC) {
    for (int i = 0; i < lista.size(); i++) {
        E objetoL = lista.get(i);
        if (objetoL.equals(objetoC)) {
            return i;
        }
    }
    return -1;
}

public boolean eliminar(E objeto) {
    Optional<E> buscar = buscar(objeto);
    E objetoE = buscar.get();
    if (objetoE != null) {
        System.out.println("Verdadero");
        return lista.remove(objetoE);
    }
    System.out.println("Falso");
    return false;
}

public boolean actualizar(E objetoA) {
    int pos = posicion(objetoA);
    if (pos >= 0) {
```

```
        lista.set(pos, objetoA);
        System.out.println("TRUE");
        return true;

    }
    System.out.println("FALSE");
    return false;
}

public abstract boolean validar(E objeto);

public abstract int generarId();

public List<E> getLista() {
    return lista;
}

public void setLista(List<E> lista) {
    this.lista = lista;
}
}

controladorActividad

package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Actividad;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author user
 */
public class controladorActividad extends AbstractControlador<Actividad> {

    private Pattern patron;
    private Matcher corpus;

    public controladorActividad(String ruta) {
        super(ruta);
    }

    @Override
    public boolean validar(Actividad objeto) {
        return true;
    }
}
```



```
@Override
public int generarId() {
    List<Actividad> temp = new ArrayList();
    for (Actividad a : super.getLista()) {
        Actividad m = (Actividad) a;
        temp.add(m);
    }

    if (temp.size() > 0 && temp != null) {
        return temp.get(temp.size() - 1).getCodigo() + 1;
    } else {
        return 1;
    }
}

public void ingresarRegex(String regex) {
    patron = Pattern.compile(regex);
}

public Set<String> obtenerUrl(String texto) {

    Set<String> resultado = new HashSet();
    corpus = patron.matcher(texto);

    while (corpus.find()) {
        resultado.add(corpus.group(0));
    }
    return resultado;
}

public List<Actividad> actividades() {

    List<Actividad> lista = new ArrayList();
    Actividad actividad;
    Iterator i = super.getLista().iterator();
    while (i.hasNext()) {
        actividad = (Actividad) i.next();
        lista.add(actividad);
    }
    return lista;
}
}
```

controladorAlumno

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Alumno;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author user
 */
public class controladorAlumno extends AbstractControlador<Alumno> {

    private controladorPersona control;

    public controladorAlumno(controladorPersona control, String ruta) {
        super(ruta);
        this.control = control;
    }

    @Override
    public boolean validar(Alumno objeto) {
        return control.validar(objeto);
    }

    @Override
    public int generarId() {
        return 0;
    }

    public List<Alumno> alumnos() {

        List<Alumno> lista = new ArrayList();
        Alumno alumno;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            alumno = (Alumno) i.next();
            lista.add(alumno);
        }
        return lista;
    }
}
```

controladorCurso

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Curso;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author user
 */
public class controladorCurso extends AbstractControlador<Curso> {

    public controladorCurso(String ruta) {
        super(ruta);
    }


    @Override
    public boolean validar(Curso objeto) {
        return true;
    }

    @Override
    public int generarId() {
        List<Curso> temp = new ArrayList();
        for (Curso a : super.getLista()) {
            Curso m = (Curso) a;
            temp.add(m);
        }

        if (temp.size() > 0 && temp != null) {
            return temp.get(temp.size() - 1).getCodigo() + 1;
        } else {
            return 1;
        }
    }

    public List<Curso> cursos() {

        List<Curso> lista = new ArrayList();
        Curso curso;
        Iterator i = super.getLista().iterator();
        while (i.hasNext()) {
            curso = (Curso) i.next();
            lista.add(curso);
        }
        return lista;
    }
}
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

}

}

controladorDocente

```
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Docente;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author user
 */
public class controladorDocente extends AbstractControlador<Docente> {

    private controladorPersona control;

    public controladorDocente(controladorPersona control, String ruta) {
        super(ruta);
        this.control = control;
    }


    @Override
    public boolean validar(Docente objeto) {
        return control.validar(objeto);
    }

    @Override
    public int generarId() {
        return 0;
    }

    public List<Docente> docentes() {

        List<Docente> lista = new ArrayList();
        Docente docente;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            docente = (Docente) i.next();
            lista.add(docente);
        }
        return lista;
    }
}
```

```
}  
}  
  
controladorPersona  
  
package ec.edu.ups.controlador;  
  
import ec.edu.ups.modelo.Persona;  
  
/**  
 *  
 * @author user  
 */  
public class controladorPersona extends AbstractControlador<Persona> {  
  
    public controladorPersona(String ruta) {  
        super(ruta);  
    }  
  
    @Override  
    public boolean validar(Persona objeto) {  
        int suma = 0;  
        String x = objeto.getCedula();  
        if (x.length() == 9) {  
            return false;  
        } else {  
            int a[] = new int[x.length() / 2];  
            int b[] = new int[(x.length() / 2)];  
            int c = 0;  
            int d = 1;  
            for (int i = 0; i < x.length() / 2; i++) {  
                a[i] = Integer.parseInt(String.valueOf(x.charAt(c)));  
                c = c + 2;  
                if (i < (x.length() / 2) - 1) {  
                    b[i] = Integer.parseInt(String.valueOf(x.charAt(d)));  
                    d = d + 2;  
                }  
            }  
  
            for (int i = 0; i < a.length; i++) {  
                a[i] = a[i] * 2;  
                if (a[i] > 9) {  
                    a[i] = a[i] - 9;  
                }  
                suma = suma + a[i] + b[i];  
            }  
            int aux = suma / 10;  
            int dec = (aux + 1) * 10;  
            if ((dec - suma) == Integer.parseInt(String.valueOf(x.charAt(x.length()  
- 1)))) {  
                return true;  
            } else if (suma % 10 == 0 && x.charAt(x.length() - 1) == '0') {
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

        return true;
    } else {
        return false;
    }
}

@Override
public int generarId() {
    return 0;
}
}

                                controladorUsuario
public class controladorUsuario extends AbstractControlador<Usuario> {

    private Usuario usuario;

    public controladorUsuario(String ruta) {
        super(ruta);
    }

    @Override
    public boolean validar(Usuario objeto) {

        return true;
    }

    @Override
    public int generarId() {
        return 0;
    }

    public List<Usuario> usuarios() {

        List<Usuario> lista = new ArrayList();
        Usuario u;
        Iterator i = super.getList().iterator();
        while (i.hasNext()) {
            u = (Usuario) i.next();
            lista.add(u);
        }
        return lista;
    }

    public Usuario getUsuario() {

```

```
        return usuario;
    }


    public boolean iniciarSesion(String correo, String pass) {

        for (Usuario usu : super.getLista()) {
            Usuario u = (Usuario) usu;
            if (u.getCorreo().equals(correo) && u.getPass().equals(pass)) {
                this.usuario = u;
                return true;
            }
        }
        return false;
    }

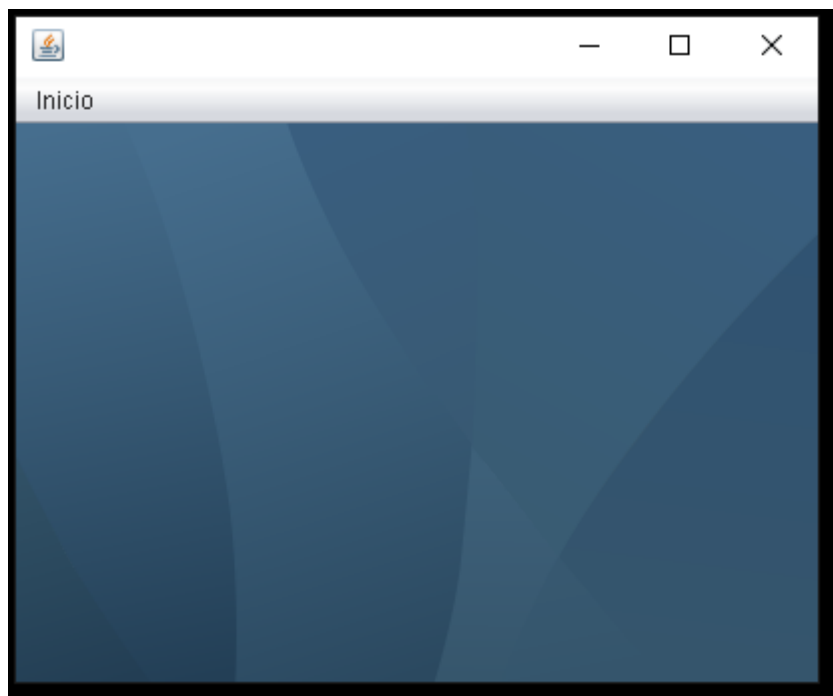
    public Usuario buscarU(Docente docente) {

        for (Usuario usu : super.getLista()) {
            Usuario u = (Usuario) usu;
            if (docente.equals(u.getDocente())) {
                this.usuario = u;
            }
        }
        return null;
    }


    }
```

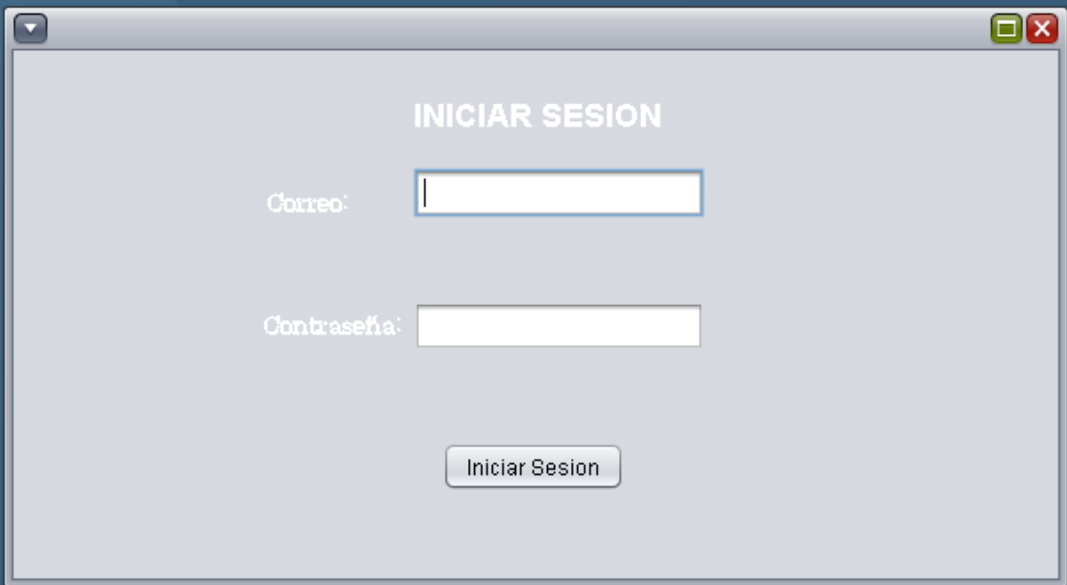
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Ventana Inicio

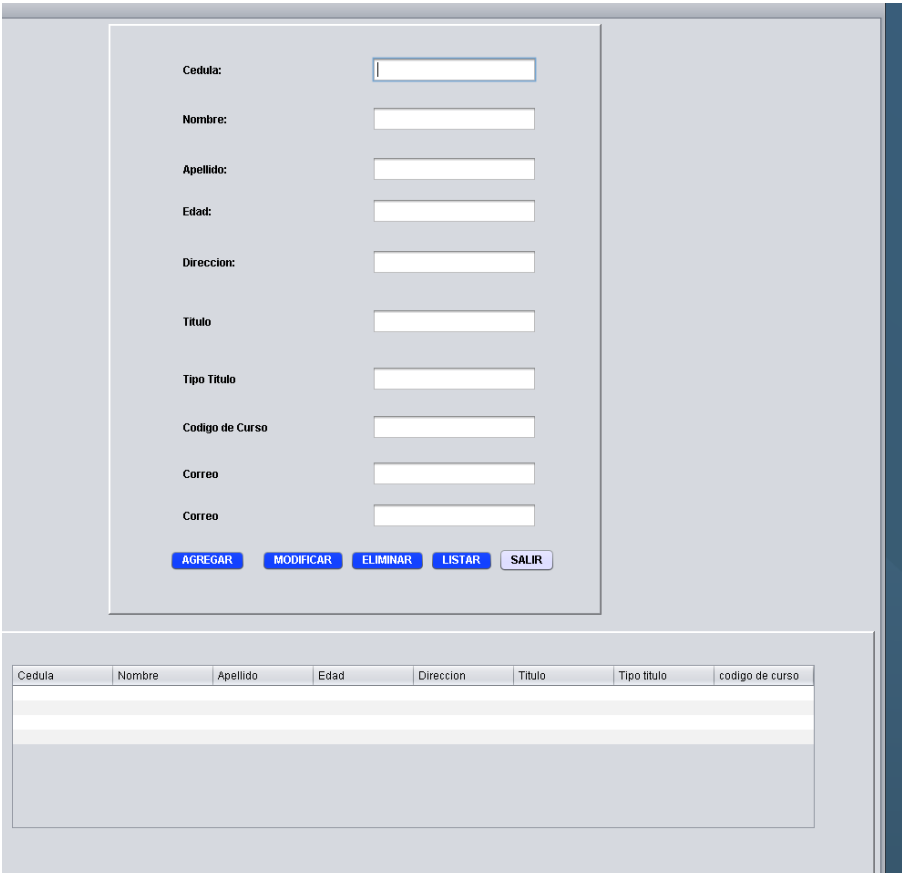



Ventana Iniciar Sesión

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

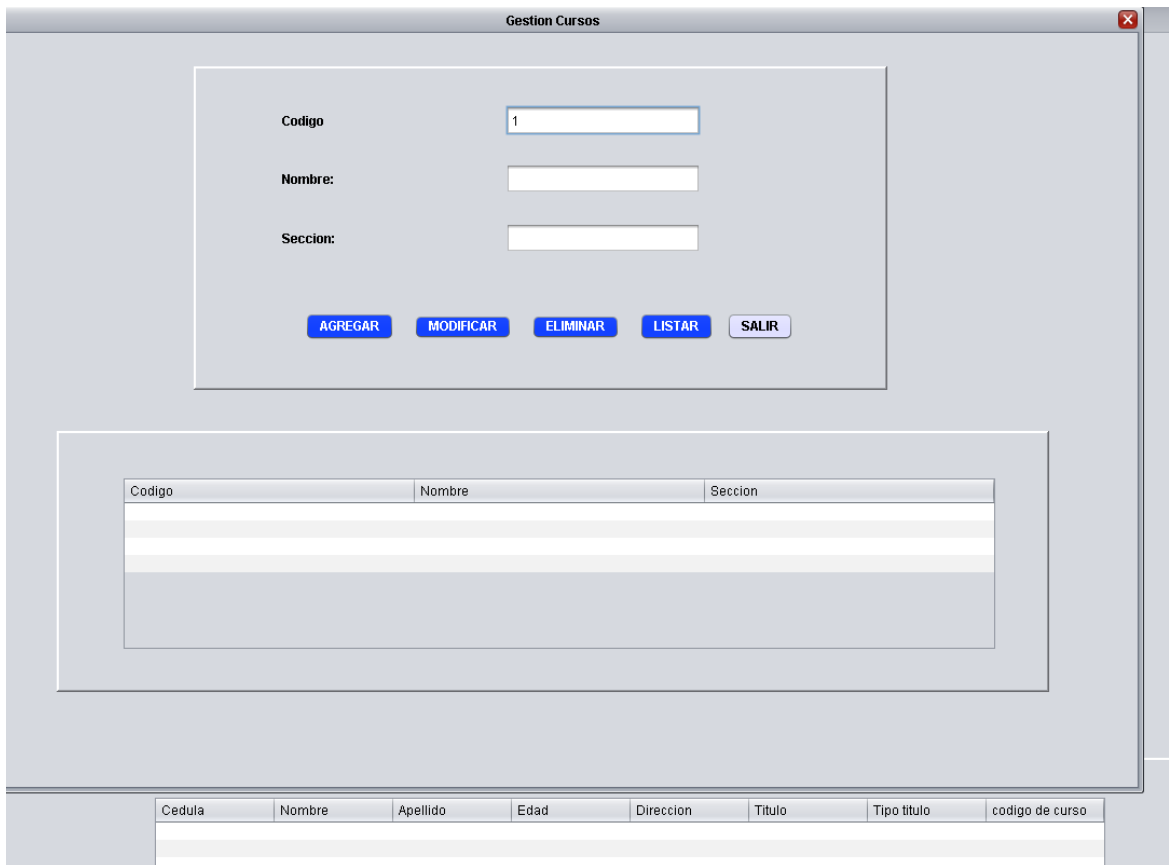


Ventana para gestion de docentes



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Ventana para gestión de cursos



Codigo	Nombre	Seccion

Cedula	Nombre	Apellido	Edad	Direccion	Titulo	Tipo titulo	codigo de curso

El programa funciona con el rector que solo gestiona a los docentes y cursos, por lo tanto, existen otros usuarios que son los docentes que solo gestiona las actividades y a los estudiantes

Cedula:

Nombre:

Apellido:

Edad:

Direccion:

Codigo de Curso:

Aula:

AGREGAR

MODIFICAR

ELIMINAR

LISTAR


SALIR

Cedula	Nombre	Apellido	Edad	Direccion	codigo de Curso	Aula

Ventana para registro de estudiantes

Ventana para el patrón de búsqueda mediante expresiones regulares

[illegible]

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

RESULTADOS OBTENIDOS:

Se logro desarrollar una aplicación con conocimientos de Java 8, programación genérica, programación orientada a objetos, MVC, reflexión y expresiones regulares para patrón de búsqueda

CONCLUSIONES:

Se dio a conocer nuevos métodos de programación para el acoplamiento y reducción de errores, con conocimientos previos de programación genérica y expresiones regulares

RECOMENDACIONES:

Investigar aplicaciones relacionadas para tener nuevos conceptos a ser aplicados en nuevos proyectos

Nombre de estudiante: JUAN JOSE CORDOVA CALLE

Firma de estudiante:

