

## FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES

**CARRERA:** COMPUTACIÓN

**ASIGNATURA:** Programación Aplicada

**NRO. PRÁCTICA:**

1

**TÍTULO PRÁCTICA:** Proyecto Interciclo

### OBJETIVO:

- Reforzar los conocimientos adquiridos en clase sobre la programación aplicada (POO, Interfaz gráfica, etc) en un contexto real.

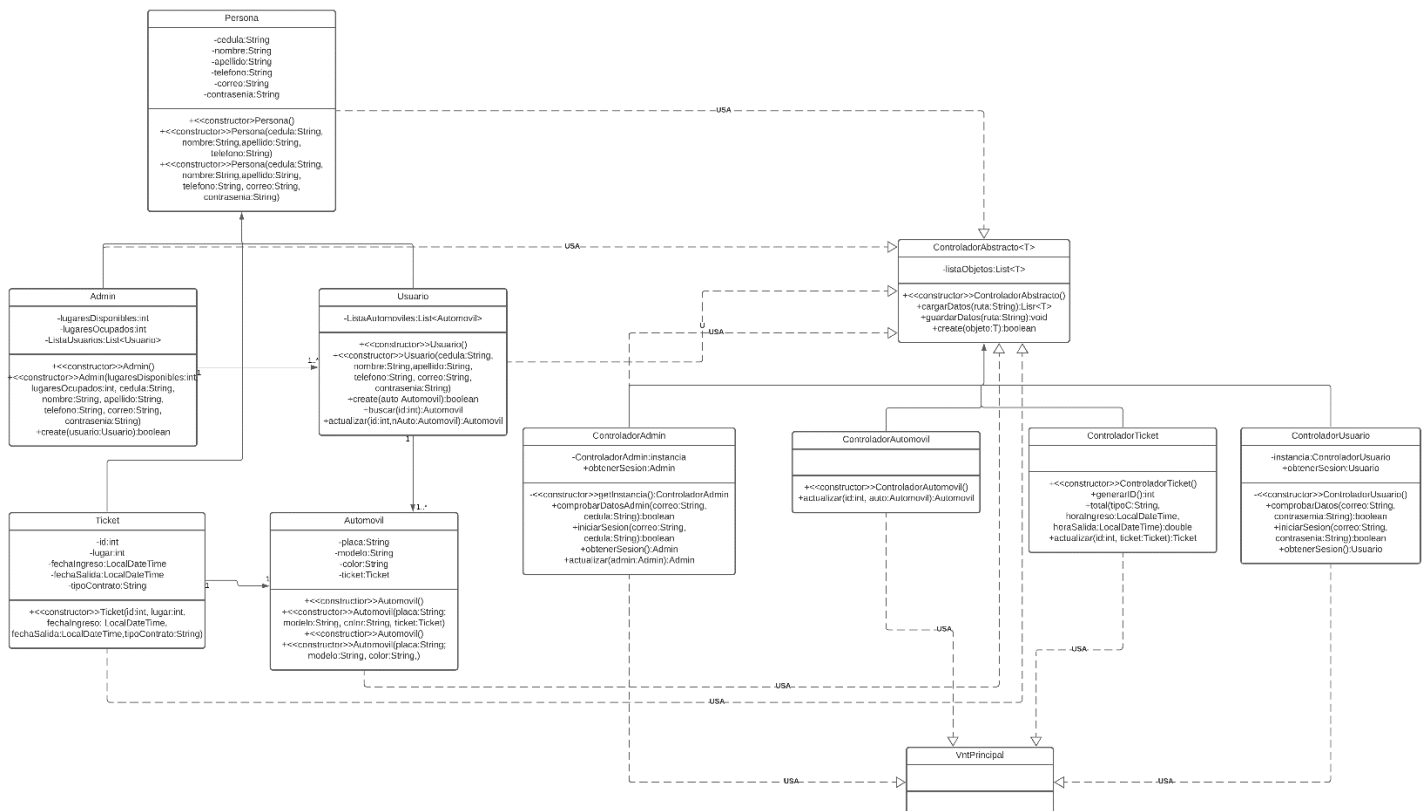
**INSTRUCCIONES** (Detallar las instrucciones que se dará al estudiante):


- Revisar los conceptos fundamentales de Java
- Establecer las características de Java basados en patrones de diseño
- Implementar y diseñar los nuevos patrones de Java
- Realizar el informe respectivo según los datos solicitados.

### ACTIVIDADES POR DESARROLLAR

#### 1. Diagrama UML

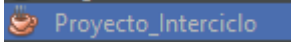
[https://lucid.app/lucidchart/c4b71f58-a1b3-4262-8489-d9956bbe3ce6/view?page=0\\_0#?folder\\_id=home&brower=icon](https://lucid.app/lucidchart/c4b71f58-a1b3-4262-8489-d9956bbe3ce6/view?page=0_0#?folder_id=home&brower=icon)



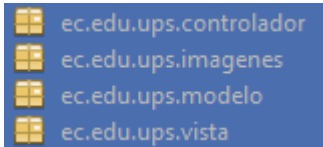
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

## 2. Creación de paquetes y controladores

### 2.1. Creamos un Proyecto en el NetBeans con el nombre de Examen\_Interciclo

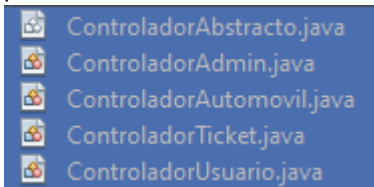


### 2.2. Creamos los paquetes para crear las clases y empezar con el proyecto nuevo creado, creamos los paquetes controladores, modelo y vista.

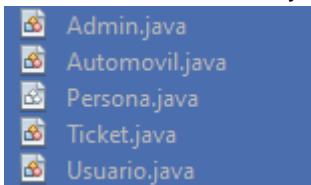


### 2.3. Dentro de cada paquete creamos las clases respectivas

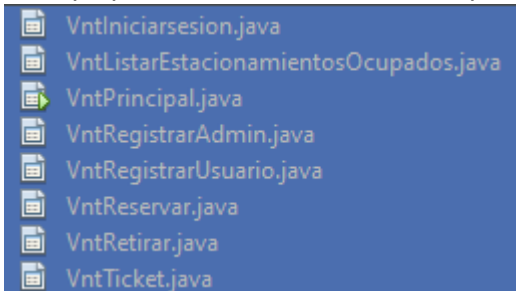
#### 2.3.1. En el paquete controlador creamos las clases controladorAbstracto que será la clase padre de las clases ControladorAbstracto, ControladorAdmin, ControladorAutomovil, ControladorTicket, ControladorUsuario así hacemos uso de lo que es programación genérica, los controladores heredaran los atributos y métodos de su clase padre.



#### 2.3.2. En el paquete modelo creamos la clase abstracta Persona, que heredara sus atributos a sus clases hijas, que son: Admin, Usuario, Ticket, y la clase Automóvil no hereda de la clase Persona.



#### 2.3.3. En el paquete interfaz se encontrará la parte de las ventanas para demostrar el funcionamiento del programa.



## 3. En cada clase creamos sus atributos, constructores, si usan métodos como los métodos los getters, setters, método equals y hashCode.

### 3.1. Persona: Esta es la clase padre de las clases Admin, usuario y ticket

```
public abstract class Persona implements Serializable{
    private String cedula;
    private String nombre;
    private String apellido;
    private String telefono;
    private String correo;
    private String cotrasenia;
```

```
public Persona(String cedula, String nombre, String apellido, String telefono, String
correo, String cotrasenia) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.correo = correo;
    this.cotrasenia = cotrasenia;
    this.telefono = telefono;
}

public Persona(String cedula, String nombre, String apellido, String telefono) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.telefono = telefono;
}

public Persona() {
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public String getNombre() {
    return nombre;
}

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

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

public String getCorreo() {
    return correo;
}
```

```
}

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

public String getCotrasenia() {
    return cotrasenia;
}

public void setCotrasenia(String cotrasenia) {
    this.cotrasenia = cotrasenia;
}


@Override
public int hashCode() {
    int hash = 3;
    hash = 47 * hash + Objects.hashCode(this.cedula);
    hash = 47 * hash + Objects.hashCode(this.correo);
    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 Persona other = (Persona) obj;
    if (!Objects.equals(this.cedula, other.cedula)) {
        return false;
    }
    if (!Objects.equals(this.correo, other.correo)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Persona ---> " + "cedula=" + cedula + ", nombre=" + nombre + ", apellido=" +
    apellido + ", telefono " + telefono + ", correo=" + correo + ", cotrasenia=" + cotrasenia + '}';
}

}

3.2. Admin
public class Admin extends Usuario implements Serializable{
    private int lugaresDisponibles;
    private int lugaresOcupados;
```

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

```

private List<Usuario> listaUsuarios;

public Admin() {
    this.lugaresDisponibles = 40;
    this.lugaresOcupados = 0;
    this.listaUsuarios = new ArrayList<>();
}

public Admin(int lugaresDisponibles, int lugaresOcupados, String cedula, String nombre,
String apellido, String telefono, String correo, String cotrasenia) {
    super(cedula, nombre, apellido, telefono, correo, cotrasenia);
    this.lugaresDisponibles = 40;
    this.lugaresOcupados = 0;
    this.listaUsuarios = new ArrayList<>();
}

public int getLugares() {
    return lugaresDisponibles;
}

public void setLugares(int lugares) {
    this.lugaresDisponibles = lugares;
}

public List<Usuario> getListaUsuarios() {
    return listaUsuarios;
}


public void setListaUsuarios(List<Usuario> listaUsuarios) {
    this.listaUsuarios = listaUsuarios;
}

public boolean create(Usuario usuario){
    return listaUsuarios.add(usuario);
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 71 * hash + this.lugaresDisponibles;
    return hash;
}

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

```

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

```

    final Admin other = (Admin) obj;
    return this.lugaresDisponibles == other.lugaresDisponibles;
}

@Override
public String toString() {
    return super.toString() + "\n---> Admin ---> " + "lugares=" + lugaresDisponibles + '}';
}
}

```

### 3.3. Usuario

```

public class Usuario extends Persona implements Serializable{
    // private int id;
    private List<Automovil> listaAutomoviles;

    public Usuario() {
        this.listaAutomoviles = new ArrayList<>();
    }

    public Usuario(String cedula, String nombre, String apellido, String telefono, String correo, String cotrasenia) {
        super(cedula, nombre, apellido, telefono, correo, cotrasenia);
        this.listaAutomoviles = new ArrayList<>();
    }

    public List<Automovil> getListaAutomoviles() {
        return listaAutomoviles;
    }


    public void setListaAutomoviles(List<Automovil> listaAutomoviles) {
        this.listaAutomoviles = listaAutomoviles;
    }

    public boolean create(Automovil auto){
        return listaAutomoviles.add(auto);
    }

    public Automovil buscar(int id){
        for (int i = 0; i < listaAutomoviles.size(); i++) {
            Automovil get = listaAutomoviles.get(i);
            if (id == get.getTicket().getId()) {
                return get;
            }
        }
        return null;
    }

    public Automovil actualizar(int id, Automovil nAuto){
        for (int i = 0; i < listaAutomoviles.size(); i++) {
            Automovil auto = listaAutomoviles.get(i);
            if (auto.getTicket().getId() == id) {
                return listaAutomoviles.set(i, nAuto);
            }
        }
    }
}

```

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

```

    return null;
}

@Override
public String toString() {
    return super.toString() + "\n---> Usuario ---> " + "listaAutomoviles=" +
listaAutomoviles + '}';
}

}

3.4. Ticket
public class Ticket extends Persona implements Serializable{
    private int id;
    private int lugar;
    private LocalDateTime fechaIngreso;
    private LocalDateTime fechaSalida;
    private String tipoContrato;
    //    private Automovil ticketAuto;

    public Ticket() {

    }

    public Ticket(int id, int lugar, LocalDateTime fechaIngreso, String tipoContrato, String
cedula, String nombre, String apellido, String telefono) {
        super(cedula, nombre, apellido, telefono);
        this.id = id;
        this.lugar = lugar;
        this.fechaIngreso = fechaIngreso;
        this.tipoContrato = tipoContrato;
    }

    public Ticket(int id, int lugar, LocalDateTime fechaIngreso, LocalDateTime fechaSalida,
String tipoContrato) {
        this.id = id;
        this.lugar = lugar;
        this.fechaIngreso = fechaIngreso;
        this.fechaSalida = fechaSalida;
        this.tipoContrato = tipoContrato;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getLugar() {
        return lugar;
    }

    public void setLugar(int Lugar) {
        this.lugar = Lugar;
    }

```

```
}

public LocalDateTime getFechaIngreso() {
    return fechaIngreso;
}

public void setFechaIngreso(LocalDateTime fechaIngreso) {
    this.fechaIngreso = fechaIngreso;
}

public LocalDateTime getFechaSalida() {
    return fechaSalida;
}

public void setFechaSalida(LocalDateTime fechaSalida) {
    this.fechaSalida = fechaSalida;
}

public String getTipoContrato() {
    return tipoContrato;
}


public void setTipoContrato(String tipoContrato) {
    this.tipoContrato = tipoContrato;
}

// public Automovil getTicketAuto() {
//     return ticketAuto;
// }
//
// public void setTicketAuto(Automovil ticketAuto) {
//     this.ticketAuto = ticketAuto;
// }

@Override
public int hashCode() {
    int hash = 7;
    hash = 37 * hash + this.id;
    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 Ticket other = (Ticket) obj;
    if (this.id != other.id) {
```



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

```

        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Ticket{" + "id=" + id + ", lugar=" + lugar + ", fechaIngreso=" + fechaIngreso + ",
fechaSalida=" + fechaSalida + ", tipoContrato=" + tipoContrato + '}';
}

}

```

### 3.5. Automovil

```

public class Automovil implements Serializable{
    private String placa;
    private String modelo;
    private String color;
    private Ticket ticket;

    public Automovil() {}

    public Automovil(String placa, String modelo, String color) {
        this.placa = placa;
        this.modelo = modelo;
        this.color = color;
    }

    public Automovil(String placa, String modelo, String color, Ticket ticket) {
        this.placa = placa;
        this.modelo = modelo;
        this.color = color;
        this.ticket = ticket;
    }

    public String getPlaca() {
        return placa;
    }


    public void setPlaca(String placa) {
        this.placa = placa;
    }

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public String getColor() {
        return color;
    }
}

```

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

```

public void setColor(String color) {
    this.color = color;
}

public Ticket getTicket() {
    return ticket;
}

public void setTicket(Ticket ticket) {
    this.ticket = ticket;
}

@Override
public int hashCode() {
    int hash = 7;
    hash = 29 * hash + Objects.hashCode(this.placa);
    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 Automovil other = (Automovil) obj;
    if (!Objects.equals(this.placa, other.placa)) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "\n---> Automovil ---> " + "placa=" + placa + ", modelo=" + modelo + ", color=" +
    color + ", ticket=" + ticket + '}';
}
}

```

4. En las clases del paquete controlador todos heredarán de la clase ControladorAbstracto y tendrán métodos para la creación de un administrador, usuarios, vehículos y tickets. En algunos métodos se aplicó lo visto en clases streams.


4.1. ControladorAbstracto: Tendrá los métodos que heredarán sus clases hijas

```

public abstract class ControladorAbstracto<T> {
    private List<T> listaObjetos;

    public ControladorAbstracto() {
        this.listaObjetos = new ArrayList<>();
    }
    // listaObjetos = new ArrayList<>();
}

```

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

```

public List<T> cargarDatos(String ruta) throws FileNotFoundException, IOException,
ClassNotFoundException{
    FileInputStream archivo = new FileInputStream(ruta);
    ObjectInputStream datos = new ObjectInputStream(archivo);
    return listaObjetos = (List<T>) datos.readObject();
}

public void guardarDatos(String ruta) throws FileNotFoundException, IOException{
    FileOutputStream archivo = new FileOutputStream(ruta);
    ObjectOutputStream datos = new ObjectOutputStream(archivo);
    datos.writeObject(listaObjetos);
}

public boolean create(T objeto){
    return listaObjetos.add(objeto);
}

public List<T> getListaObjetos() {
    return listaObjetos;
}

public void setListaObjetos(List<T> listaObjetos) {
    this.listaObjetos = listaObjetos;
}
}

```

#### 4.2. ControladorAdmin

```

public class ControladorAdmin extends ControladorAbstracto<Admin> {


    private static ControladorAdmin instancia;

    private ControladorAdmin() {
        super();
    }

    public static ControladorAdmin getInstancia() {
        if (instancia == null) {
            instancia = new ControladorAdmin();
        }
        return instancia;
    }

    /**
     *
     * @param correo
     * @param cedula
     * @return
     */
    public boolean comprobarDatosAdmin(String correo, String cedula) {
        return getListaObjetos().stream().filter(c -> correo.equals(c.getCorreo()) &&
cedula.equals(c.getCedula()))
                .noneMatch(c1 -> c1.getCorreo().equals(correo) &&
c1.getCedula().equals(cedula));
    }
}

```

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

```

Admin obtenerSesion;
/**
 *
 * @param correo
 * @param contrasenia
 * @return
 */
public boolean iniciarSesion(String correo, String contrasenia) {
    for (int i = 0; i < getListaObjetos().size(); i++) {
        var admin = getListaObjetos().get(i);
        if (correo.equals(admin.getCorreo()) && contrasenia.equals(admin.getCotrasenia())) {
            obtenerSesion = admin;
            return true;
        }
    }
    return false;
}

public Admin obtenerSesion(){
    return obtenerSesion;
}

public Admin actualizar(Admin admni){
    return getListaObjetos().set(0, admni);
}

// public static Admin registrar() {
//     if (admin == null) {
//         usuario = new Usuario();
//     }
//     return usuario;
// }

4.3. ControladorUsuario
public class ControladorUsuario extends ControladorAbstracto<Usuario> {

//     private static Usuario usuario;
//     private static ControladorUsuario instancia;

private ControladorUsuario() {
    super();
}

public static ControladorUsuario getInstancia() {
    if (instancia == null) {
        instancia = new ControladorUsuario();
    }
    return instancia;
}

/**
 *
 * @param correo

```

```

* @param cedula
* @param ruta
* @return
* @throws IOException
* @throws FileNotFoundException
* @throws ClassNotFoundException
*/
public boolean comprobarDatos(String correo, String cedula){
    return getListaObjetos().stream().filter(c -> correo.equals(c.getCorreo()) &&
cedula.equals(c.getCedula()))
        .noneMatch(c1 -> c1.equals(correo) && c1.equals(cedula));
}

Usuario obtenerSesion;
/**
 *
 * @param correo
 * @param contrasenia
 * @return
 */
public boolean iniciarSesion(String correo, String contrasenia) {
    for (int i = 0; i < getListaObjetos().size(); i++) {
        var admin = getListaObjetos().get(i);
        if (correo.equals(admin.getCorreo()) && contrasenia.equals(admin.getCotrasenia())) {
            obtenerSesion = admin;
            return true;
        }
    }
    return false;
}

public Usuario obtenerSesion(){
    return obtenerSesion;
}

// public static Usuario registrar() {
//     if (usuario == null) {
//         usuario = new Usuario();
//     }
//     return usuario;
// }

4.4. ControladorAutomovil
public class ControladorAutomovil extends ControladorAbstracto<Automovil>{

    public ControladorAutomovil() {
        super();
    }


    public Automovil actualizar(int id, Automovil auto) {
        for (int i = 0; i < getListaObjetos().size(); i++) {
            var get = getListaObjetos().get(i);
            if (id == get.getTicket().getId()) {
                return getListaObjetos().set(id, auto);
            }
        }
    }

```

```
}  
return null;  
}
```

#### 4.5. ControladorTicket

```
public class ControladorTicket extends ControladorAbstracto<Ticket> {  
  
    public ControladorTicket() {  
        super();  
    }  
  
    public int generarID() {  
        if (!getListaObjetos().isEmpty()) {  
            return getListaObjetos().size();  
        }  
        return 0;  
    }  
  
    public double total(String tipoC, LocalDateTime horaIngreso, LocalDateTime horaSalida) {  
        double pagar = 0;  
        if (tipoC.equals("Por Horas")) {  
            long nhoras = ChronoUnit.HOURS.between(horaIngreso, horaSalida);  
            if (nhoras <= 12) {  
                if (nhoras == 0) {  
                    pagar = 0.50;  
                } else {  
                    pagar = nhoras * 0.50;  
                }  
            } else {  
                pagar = (nhoras * 0.50) * 0.1 + (nhoras * 0.50);  
            }  
            return pagar;  
        } else if (tipoC.equals("Por Dias")) {  
            long nDias = ChronoUnit.DAYS.between(horaIngreso, horaSalida);  
            if (nDias <= 10) {  
                if (nDias == 0) {  
                    pagar = 5.0;  
                } else {  
                    pagar = nDias * 5;  
                }  
            } else {  
                pagar = (nDias * 5) * 0.10 + (nDias * 5);  
            }  
            return pagar;  
        } else if (tipoC.equals("Por Semanas")) {  
            long nSemanas = ChronoUnit.WEEKS.between(horaIngreso, horaSalida);  
            if (nSemanas <= 4) {  
                if (nSemanas == 0) {  
                    pagar = 30.0;  
                } else {  
                    pagar = nSemanas * 30;  
                }  
            } else {  
                pagar = (nSemanas * 30) * 0.1 + (nSemanas * 30);  
            }  
        }  
    }  
}
```

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

```

    }
    return pagar;
} else if (tipoC.equals("Por Mes")) {
    long nMes = ChronoUnit.MONTHS.between(horaIngreso, horaSalida);
    if (nMes > 1) {
        pagar = (nMes * 60) * 0.1 + (nMes * 60);
    } else {
        pagar = 60.0;
    }
    return pagar;
}
if (true) {

}
return 0;
}

public Ticket actualizar(int id, Ticket ticket) {
    for (int i = 0; i < getListaObjetos().size(); i++) {
        var get = getListaObjetos().get(i);
        if (id == get.getId()) {
            return getListaObjetos().set(id, ticket);
        }
    }
    return null;
}
}


```

## 5. Funcionamiento de la aplicación

### 5.1. VntPrincipal

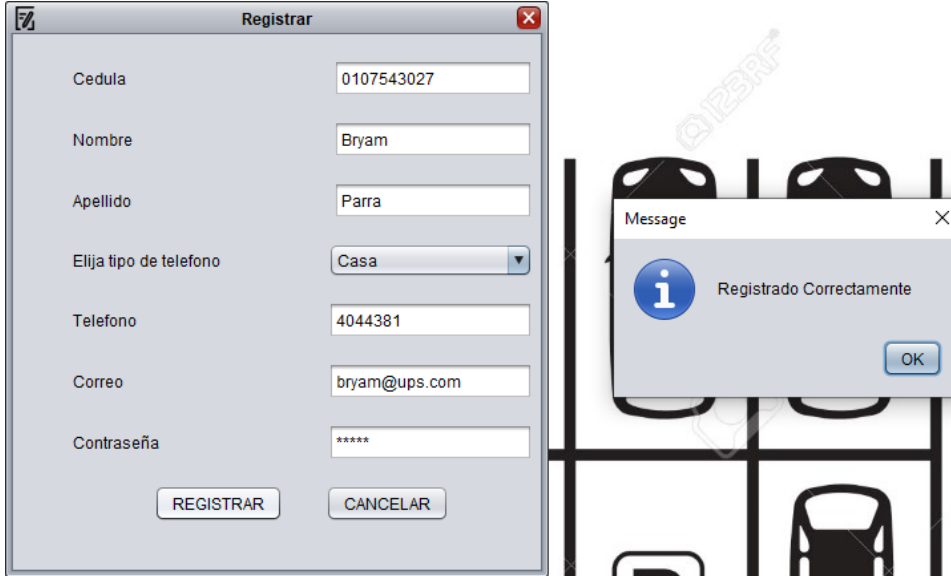
Las ventanas tendrán un menú el cual se desplegara, en el menú Registrar Administrador, se registrara un administrador



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

## 5.2. VntRegistrarAdministrador


En esta ventana se podrá crear un administrador, en los campos de la cedula, teléfono, correo validan que sean correctos los datos, serán validados aplicando expresiones regulares.



The screenshot shows the 'Registrar' window for administrators. The fields are filled with: Cedula (0107543027), Nombre (Bryam), Apellido (Parra), Elija tipo de telefono (Casa), Telefono (4044381), Correo (bryam@ups.com), and Contraseña (\*\*\*\*\*). The 'REGISTRAR' button is highlighted. To the right, a 'Message' dialog box displays 'Registrado Correctamente' with an 'OK' button.

## 5.3. VntRegistrarUsuario

En esta ventana se crearan el usuario, validara que se ingresen bien los campos cedula, correo y telefono




The screenshot shows the 'Registrar' window for users. The fields are filled with: Cedula (0107543019), Nombre (julian), Apellido (zambrano), Elija tipo de telefono (Movil), Telefono (0994957717), Correo (julian@ups.com), and Contraseña (\*\*\*\*\*). The 'REGISTRAR' button is highlighted. To the right, a 'Message' dialog box displays 'Registrado Correctamente' with an 'OK' button.

## 5.4. VntIniciarSesion

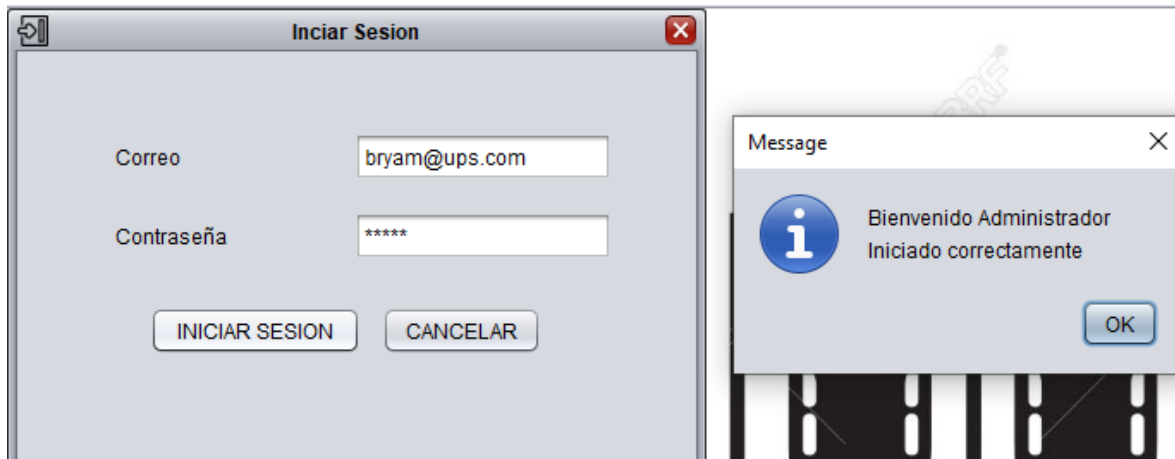
Cuando el administrador inicia sesión el en el menú se visualizará la opción de registrar a los usuarios.



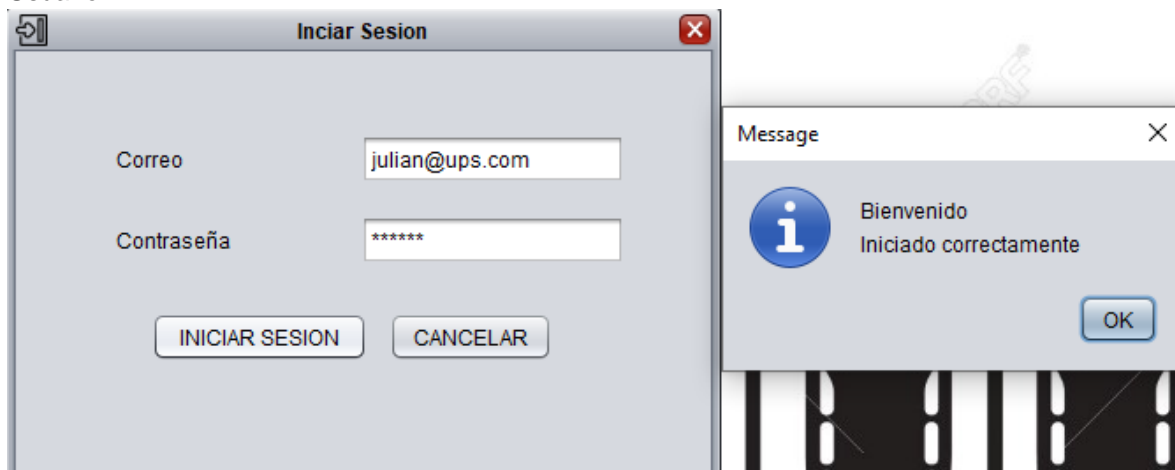
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


Cuando el usuario inicia sesión se habilitará un nuevo menú llamado gestión en el que podrá reservar un estacionamiento y generar un ticket.

Administrador



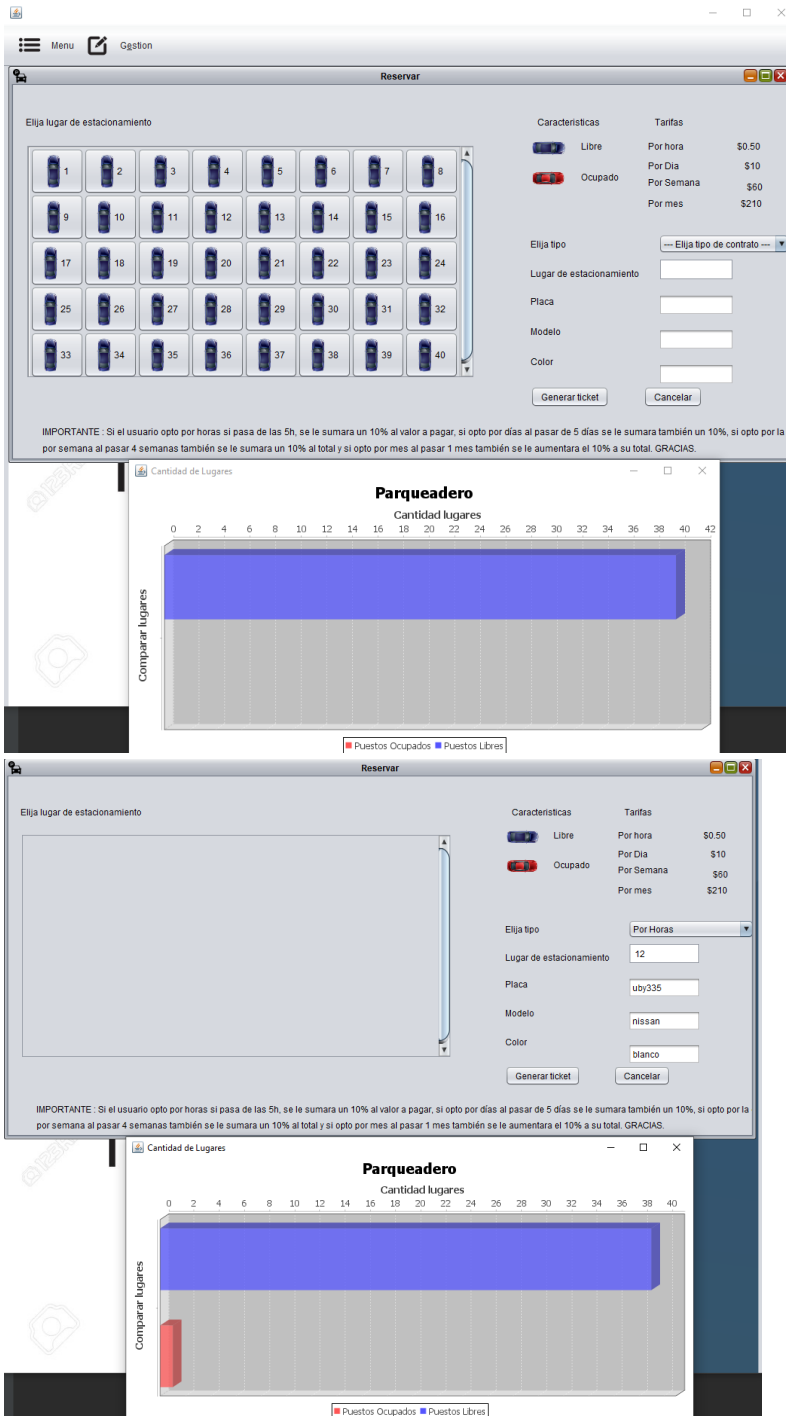
Usuario



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

### 5.5. VntReservar

Al Iniciar la ventana visualizar un grafico de barra en azul que son la cantidad de lugares disponibles y los que ya están reservados y existirá la opción de elegir el lugar donde se va a estacionar, tendrá datos informando las tarifas del parqueadero y un aviso acerca del tiempo que desea reservar el lugar.




The screenshot displays the 'Reservar' (Reserve) window of the VntReservar application. The interface includes a grid of 40 parking spots (numbered 1-40) and a bar chart titled 'Parqueadero' showing the number of available (blue) and occupied (red) spots. The 'Tarifas' (Rates) table is visible, showing rates for different reservation periods.

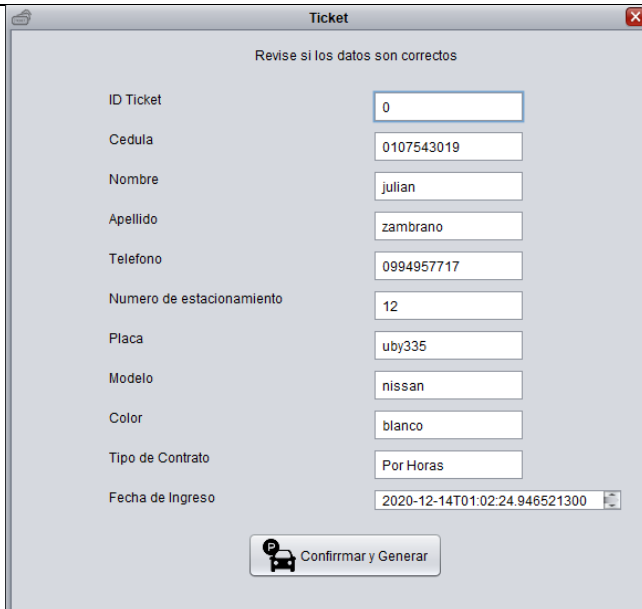
Características	Tarifas
Libre	Por hora \$0.50
Ocupado	Por Día \$10
	Por Semana \$60
	Por mes \$210

The 'Reservar' window also includes fields for 'Elija lugar de estacionamiento' (Select parking location), 'Elija tipo' (Select type), 'Lugar de estacionamiento' (Parking location), 'Placa' (Plate), 'Modelo' (Model), and 'Color' (Color). The 'Generar ticket' (Generate ticket) button is visible at the bottom.

Below the 'Reservar' window, the 'Cantidad de Lugares' (Number of Spaces) window is shown, displaying a bar chart with the title 'Parqueadero' and the subtitle 'Cantidad lugares'. The chart shows the number of available (blue) and occupied (red) spaces. The legend indicates 'Puestos Ocupados' (Occupied Spaces) in red and 'Puestos Libres' (Free Spaces) in blue.

Al hacer clic en generar ticket abrirá una ventana en el que visualizara los datos que tendrá el ticket con los datos del usuario, el auto y del ticket.

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



**Ticket**

Revise si los datos son correctos

ID Ticket: 0

Cedula: 0107543019

Nombre: julian

Apellido: zambrano

Telefono: 0994957717

Numero de estacionamiento: 12

Placa: uby335

Modelo: nissan

Color: blanco

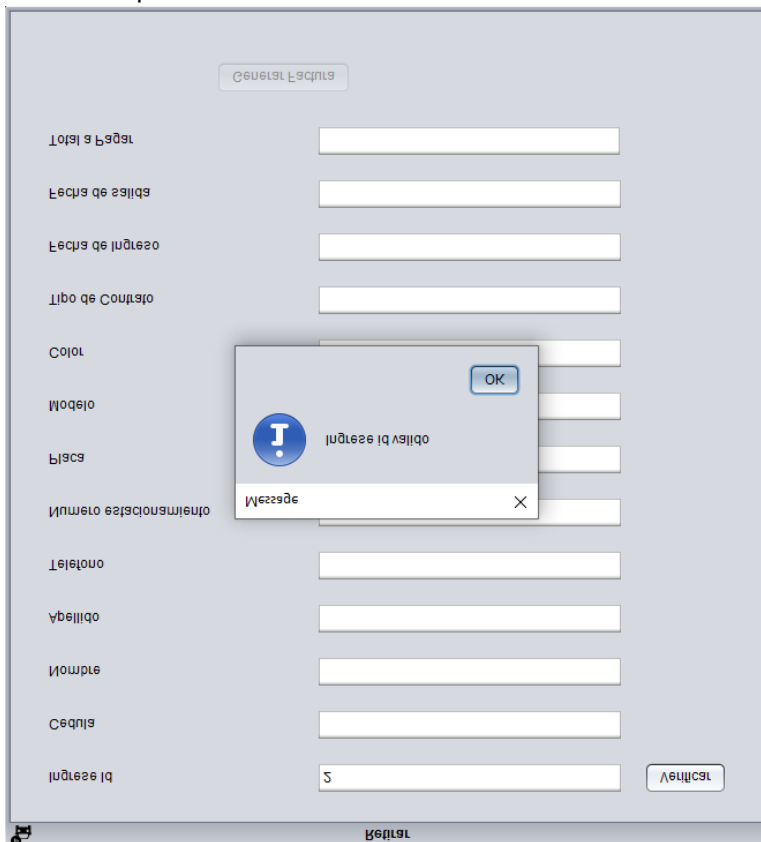
Tipo de Contrato: Por Horas

Fecha de Ingreso: 2020-12-14T01:02:24.946521300

Confirmar y Generar

## 5.6. VntRetirar

En esta ventana el usuario accederá para retirar su automóvil y pagar la cantidad según la opción que escogió, por horas, días, semanas y mes. Solo ingresa el id para buscar el ticket, si no existe ese ticket, se visualizara mensaje indicando que ese id no existe.



**VntRetirar**

Generar Factura

Total a Pagar:

Fecha de salida:

Fecha de ingreso:

Tipo de Contrato:

Color:

Modelo:

Placa:

Numero estacionamiento:

Telefono:

Apellido:

Nombre:


Cedula:

Ingrese id: 5

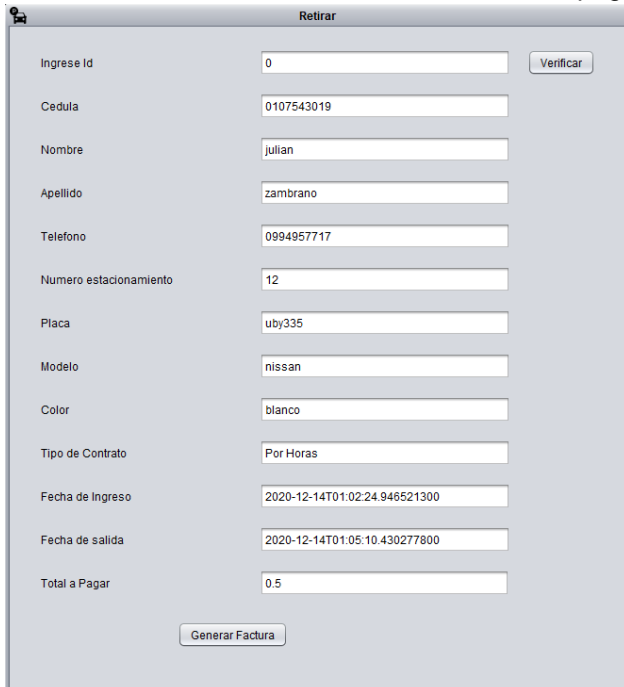
Verificar

Message: Ingrese id valido

OK

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

Visualizara los datos de la factura con el valor a pagar por la reserva del lugar de estacionamiento



**Retirar**

Ingrese Id: 0

Cedula: 0107543019

Nombre: Julian

Apellido: zambrano

Telefono: 0994957717

Numero estacionamiento: 12

Placa: uby335

Modelo: nissan

Color: blanco

Tipo de Contrato: Por Horas

Fecha de Ingreso: 2020-12-14T01:02:24.946521300

Fecha de salida: 2020-12-14T01:05:10.430277800

Total a Pagar: 0.5

#### 5.7. VntListarEstacionamiento

Para esta opción solo el administrador puede acceder y podrá visualizar a los usuarios registrados, los vehículos registrados y sus tickets generados



**Listar**


Cedula	Nombre	Apellido	Telefono	Correo	Contraseña
0107543019	Julian	zambrano	0994957717	julian@ups.com	julian

ID	Lugar	Tipo Contrato	Placa	Modelo	Color	Fecha Ingreso	Fecha Salida
0	12	Por Horas	uby335	nissan	blanco	2020-12-14T01...	2020-12-14T01...

#### RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

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

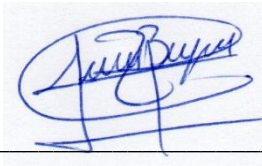
#### CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos. Los estudiantes implementan soluciones gráficas en sistemas.

#### RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica. Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la práctica.**

**Nombre de estudiante:** Bryam Parra



**Firma de estudiante:** \_\_\_\_\_