

	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: Proyecto Interciclo	
OBJETIVO: <ul style="list-style-type: none"> 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):		1. Revisar los conceptos fundamentales de Java	
		2. Establecer las características de Java basados en patrones de diseño	
		3. Implementar y diseñar los nuevos patrones de Java	
		4. Realizar el informe respectivo según los datos solicitados.	
ACTIVIDADES POR DESARROLLAR			

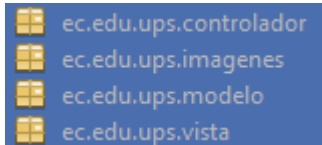
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

1. Creación de paquetes y controladores

1.1. Creamos un Proyecto en el NetBeans con el nombre de Examen_Interciclo

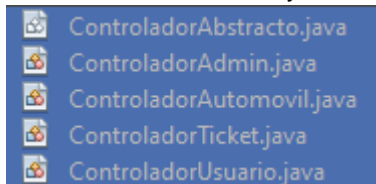


1.2. Creamos los paquetes para crear las clases y empezar con el proyecto nuevo creado, creamos los paquetes controladores, modelo y vista.

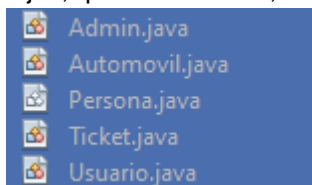


1.3. Dentro de cada paquete creamos las clases respectivas

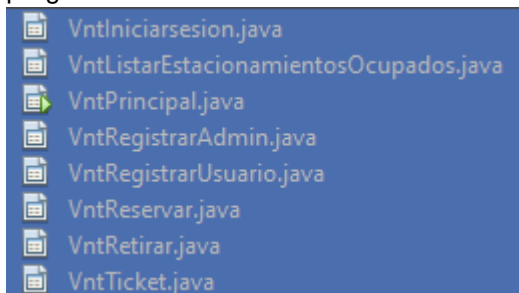
1.3.1. En el paquete controlador creamos las clases controladorAbstracto que será la clase padre de las clases ControladorAbstracto, ControladorAdmin, ControladorAutomovil, ControladorTicket, ControladorUsuario así hacemos uso de lo que es programación genérica, los controladores heredaran los atributos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



1.3.2. 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.



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



2. Creamos las Entity class, y relacionamos con sus respectivas clases.

2.1. Admin

```
@Entity
//@NamedQueries({
    @NamedQuery(name = "buscarCedula", query = "Select a from AdminP a where a.cedula = :cedula AND " + "a.correo = :correo" )
    @NamedQuery(name = "iniciarA", query = "Select a from AdminP a where a.correo = :correo AND " + "a.cotrasenia = :cotrasenia" )
})
```

```
/**})  
public class AdminP implements Serializable {  
  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private Long id;  
    @Column(name = "Cedula")  
    private String cedula;  
    @Column(name = "Nombre")  
    private String nombre;  
    @Column(name = "Apellido")  
    private String apellido;  
    @Column(name = "Telefono")  
    private String telefono;  
    @Column(name = "Correo")  
    private String correo;  
    @Column(name = "Contraseña")  
    private String cotrasenia;  
    @Column(name = "Lugares_Disponibles")  
    private int lugaresDisp;  
    @Column(name = "Lugares_Ocupados")  
    private int lugaresOcupados;  
    ///   
    @OneToMany(mappedBy = "admin", cascade = CascadeType.ALL, fetch =  
FetchType.EAGER)  
    private List<Usuario> listaUsuarios;  
    ///   
    @OneToMany(mappedBy = "admin", cascade = CascadeType.ALL, fetch =  
FetchType.EAGER)  
    private List<AutomovilP> listaAutomoviles;  
  
    public AdminP() {  
        listaUsuarios = new ArrayList<>();  
        listaAutomoviles = new ArrayList<>();  
    }  
  
    public AdminP(String cedula, String nombre, String apellido, String telefono,  
String correo, String cotrasenia, int lugaresDisp, int lugaresOcupados) {  
        this.cedula = cedula;  
        this.nombre = nombre;  
        this.apellido = apellido;  
        this.telefono = telefono;  
        this.correo = correo;  
        this.cotrasenia = cotrasenia;  
        this.lugaresDisp = lugaresDisp;  
        this.lugaresOcupados = lugaresOcupados;  
        listaUsuarios = new ArrayList<>();  
        listaAutomoviles = new ArrayList<>();  
    }  
  
    public AdminP(Long id, String cedula, String nombre, String apellido, String  
telefono, String correo, String cotrasenia, int lugaresDisp, int lugaresOcupados,  
List<Usuario> listaUsuarios) {
```

```
this.id = id;
this.cedula = cedula;
this.nombre = nombre;
this.apellido = apellido;
this.telefono = telefono;
this.correo = correo;
this.cotrasenia = cotrasenia;
this.lugaresDisp = lugaresDisp;
this.lugaresOcupados = lugaresOcupados;
this.listaUsuarios = new ArrayList<>();
}

public Long getId() {
    return id;
}

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

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;
}

public int getLugaresDisp() {
    return lugaresDisp;
}

public void setLugaresDisp(int lugaresDisp) {
    this.lugaresDisp = lugaresDisp;
}

public int getLugaresOcupados() {
    return lugaresOcupados;
}

public void setLugaresOcupados(int lugaresOcupados) {
    this.lugaresOcupados = lugaresOcupados;
}

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


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

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

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

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
```

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

```

public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
    set
    if (!(object instanceof AdminP)) {
        return false;
    }
    AdminP other = (AdminP) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

// @Override
// public String toString() {
//     return "ec.edu.ups.modelo.Admin[ id=" + id + " ]";
// }


@Override
public String toString() {
    return "AdminP{" + "id=" + id + ", cedula=" + cedula + ", nombre=" + nombre +
", apellido=" + apellido + ", telefono=" + telefono + ", correo=" + correo + ",
cotrasenia=" + cotrasenia + ", lugaresDisp=" + lugaresDisp + ", lugaresOcupados=" +
lugaresOcupados + ", listaUsuarios=" + listaUsuarios + ", listaAutomoviles=" +
listaAutomoviles + '}';
}

}

2.2. Usuario
@Entity
//@NamedQueries({
    @NamedQuery(name = "buscarCedulaU", query = "Select u from Usuario u where
u.cedula = :cedula AND " + " u.correo = :correo" )
    @NamedQuery(name = "IniciarU", query = "Select u from Usuario u where
u.correo = :correo AND u.cotrasenia = :cotrasenia" )
    @NamedQuery(name = "buscarticket", query = "Select u from Usuario u where
u.id = :id AND " + " u.cedula = :cedula" )
//})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "Cedula")
    private String cedula;
    @Column(name = "Nombre")
    private String nombre;
    @Column(name = "Apellido")
    private String apellido;
    @Column(name = "Telefono")
    private String telefono;
    @Column(name = "Correo")

```

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

```

private String correo;
@Column(name = "Contraseña")
private String cotrasenia;
//////////
@OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL, fetch =
FetchType.EAGER)
private List<AutomovilP> listaAutomoviles;
///
@ManyToOne
@JoinColumn(name = "fk_Admin")
private AdminP admin;
//////////

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

public Usuario(String cedula, String nombre, String apellido, String telefono,
String correo, String cotrasenia, AdminP admin) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.telefono = telefono;
    this.correo = correo;
    this.cotrasenia = cotrasenia;
    this.admin = admin;
}

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


public Long getId() {
    return id;
}

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

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {

```

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

```

        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;
    }

    public List<AutomovilP> getListAutomoviles() {
        return listaAutomoviles;
    }

    public void setListAutomoviles(List<AutomovilP> listaAutomoviles) {
        this.listaAutomoviles = listaAutomoviles;
    }

    public AdminP getAdmin() {
        return admin;
    }

```


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

```

public void setAdmin(AdminP admin) {
    this.admin = admin;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof Usuario)) {
        return false;
    }
    Usuario other = (Usuario) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

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

@Override
public String toString() {
    return "ec.edu.ups.modelo.Usuario[ id=" + id + " ]";
}
}

```


2.3. Ticket

```

@Entity
public class TicketP implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "Cedula")
    private String cedula;
    @Column(name = "Nombre")

```

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

```

private String nombre;
@Column(name = "Apellido")
private String apellido;
@Column(name = "Telefono")
private String telefono;
@Column(name = "Correo")
private String correo;
@Column(name = "Contraseña")
private String cotrasenia;
@Column(name = "lugar")
private int lugar;
@Column(name = "fecha_Ingreso")
private LocalDateTime fechaIngreso;
@Column(name = "fecha_Salida")
private LocalDateTime fechaSalida;
@Column(name = "Tipo_Contrato")
private String tipoContrato;
@Column(name = "Finalizado")
private boolean finalizadoC;

public TicketP() {
    finalizadoC = false;
}


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

public TicketP(String cedula, String nombre, String apellido, String telefono,
int lugar, LocalDateTime fechaIngreso, String tipoContrato, boolean finalizadoC) {
    this.cedula = cedula;
    this.nombre = nombre;
    this.apellido = apellido;
    this.telefono = telefono;
    this.lugar = lugar;
    this.fechaIngreso = fechaIngreso;
    this.tipoContrato = tipoContrato;
    this.finalizadoC = finalizadoC;
}

public Long getId() {
    return id;
}

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

```

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

```

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;
}

public int getLugar() {
    return lugar;
}

public void setLugar(int lugar) {

```

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

```

        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 boolean isFinalizadoC() {
        return finalizadoC;
    }

    public void setFinalizadoC(boolean finalizadoC) {
        this.finalizadoC = finalizadoC;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not
set
        if (!(object instanceof TicketP)) {
            return false;
        }
        TicketP other = (TicketP) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
    }

```

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

```

        return true;
    }

    @Override
    public String toString() {
        return "ec.edu.ups.modelo.Ticket[ id=" + id + " ]";
    }

}

```

2.4. Automóvil

```

@Entity
public class AutomovilP implements Serializable {


    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @Column(name = "placa")
    private String placa;
    @Column(name = "modelo")
    private String modelo;
    @Column(name = "color")
    private String color;
    ///
    @OneToOne
    @JoinColumn(name = "fk_ticket")
    private TicketP ticket;
    ///
    @ManyToOne
    @JoinColumn(name = "fk_admin")
    private AdminP admin;
    ///
    @ManyToOne
    @JoinColumn(name = "fk_usuario")
    private Usuario usuario;

    public AutomovilP() {
    }

    //    public AutomovilP(String placa, String modelo, String color, TicketP ticket) {
    //        this.placa = placa;
    //        this.modelo = modelo;
    //        this.color = color;
    //        this.ticket = ticket;
    //    }

    public AutomovilP(String placa, String modelo, String color, TicketP ticket,
AdminP admin, Usuario usuario) {
        this.placa = placa;
        this.modelo = modelo;
        this.color = color;
        this.ticket = ticket;
        this.admin = admin;
        this.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 Long getId() {
    return id;
}

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

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;
}

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

public TicketP getTicket() {
    return ticket;
}


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

public AdminP getAdmin() {
    return admin;
}

public void setAdmin(AdminP admin) {
    this.admin = admin;
}

public Usuario getUsuario() {
    return usuario;
}

```

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

```

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

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not
set
    if (!(object instanceof AutomovilP)) {
        return false;
    }
    AutomovilP other = (AutomovilP) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "ec.edu.ups.modelo.Automovil[ id=" + id + " ]";
}

}

```

3. 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.

3.1. ControladorAbstracto: Tendrá los métodos que heredaran sus clases hijas y el funcionamiento del hilo para simular al automóvil estacionarse en el lugar seleccionado por el usuario

```

public abstract class ControladorAbstractoP<T> {
    private List<T> listaObjetos;
    private Class<T> clase;
    private EntityManager emP;

    public ControladorAbstractoP() {
        listaObjetos = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        clase = (Class) pt.getActualTypeArguments()[0];
        emP = JPAUtilsP.getEntityManagerP();
    }

    public ControladorAbstractoP(EntityManager em) {
        listaObjetos = new ArrayList<>();
        Type t = getClass().getGenericSuperclass();
    }
}

```

```
ParameterizedType pt = (ParameterizedType) t;
clase = (Class) pt.getActualTypeArguments()[0];
this.emP = em;
}

public T create(T objeto){
    emP.getTransaction().begin();
    emP.persist(objeto);
    emP.getTransaction().commit();
    listaObjetos.add(objeto);
    return objeto;
}

// public boolean delete(T objeto){
//     emP.getTransaction().begin();
//     emP.remove(emP.merge(objeto));
//     emP.getTransaction().commit();
//     listaObjetos.remove(objeto);
//     return true;
// }

public T update(T objeto){
    emP.getTransaction().begin();
    objeto = emP.merge(objeto);
    emP.getTransaction().commit();
    this.findAll();
    return objeto;
}

// public T read(Object id){
//     return (T) emP.find(clase, id);
// }

public List<T> findAll(){
    return emP.createQuery("Select t from " + clase.getSimpleName() + "
t").getResultList();
}

public List<T> getListaObjetos() {
    return listaObjetos = emP.createQuery("Select t from " +
clase.getSimpleName() + " t").getResultList();
}

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

public Class<T> getClase() {
    return clase;
}

public void setClase(Class<T> clase) {
    this.clase = clase;
}
}
```



```
public EntityManager getEm() {
    return emP;
}

public void setEm(EntityManager em) {
    this.emP = em;
}


public synchronized void mover(JLabel boton, Thread t, int x, int y) {
    //    int xB = boton.getX();
    //    int yB = boton.getY();
    //    boton.setLocation(597, 468);
    int i;
    for (i = 651; i >= x; i--) {
        try {
            boton.setLocation(i, 468);
            //System.out.println("i == " + i);

            Thread.sleep(5);
        } catch (InterruptedException ex) {
            Logger.getLogger(ControladorAutomovilP.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    for (int j = 420; j >= y; j--) {
        try {
            boton.setLocation(i, j);
            //System.out.println("j == " + j);
            Thread.sleep(5);
        } catch (InterruptedException ex) {
            Logger.getLogger(ControladorAutomovilP.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    t.stop();
}
```

3.2. ControladorAdmin

```
public class ControladorAdminP extends ControladorAbstractoP<AdminP>{
    AdminP obtenerSesion;
    public ControladorAdminP() {
        super();
    }

    public boolean comprobarDatosAdmin(String correo, String cedula) {
        var ced = getEm().createNamedQuery("buscarCedula").setParameter("cedula",
cedula).setParameter("correo", correo).getResultList();
        System.out.println("ced " + ced);
        if (ced.isEmpty()) {
            System.out.println("ced " + ced);
            return true;
        }else{
            return false;
        }
    }
}
```

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

```

    }
}

public boolean iniciarSesionA(String correo, String contrasenia) {
    var admC = getEm().createNamedQuery("iniciarA").setParameter("correo",
correo).setParameter("cotrasenia", contrasenia).getResultList();
    if (admC.isEmpty()) {
        return false;
    }else{
        obtenerSesion = (AdminP) admC.get(0);
        return true;
    }
}

public AdminP obtenerSesion(){
    return obtenerSesion;
}

public AdminP getObtenerSesion() {
    return obtenerSesion;
}

public void setObtenerSesion(AdminP obtenerSesion) {
    this.obtenerSesion = obtenerSesion;
}

}

3.3. ControladorUsuario
public class ControladorUsuarioP extends ControladorAbstractoP<Usuario>{
    Usuario obtenerSesion;

    public ControladorUsuarioP() {
        super();
    }

    public boolean comprobarDatos(String correo, String cedula) {
        var ced = getEm().createNamedQuery("buscarCedulaU").setParameter("cedula",
cedula).setParameter("correo", correo).getResultList();
        System.out.println("cedUU " + ced);
        if (ced.isEmpty()) {
            System.out.println("cedUU " + ced);
            return true;
        }else{
            return false;
        }
    }

    public boolean iniciarSesionU(String correo, String contrasenia) {
        var admC = getEm().createNamedQuery("IniciarU").setParameter("correo",
correo).setParameter("cotrasenia", contrasenia).getResultList();
        if (admC.isEmpty()) {
            return false;

```

```

    }else{
        obtenerSesion = (Usuario) admC.get(0);
        return true;
    }
}

public List<Usuario> buscar(int id){
    List<Usuario> uc =
getEm().createNamedQuery("buscarticket").setParameter(":id", id).getResultList();
    if (uc != null && uc.get(0).getCedula().equals(obtenerSesion().getCedula()))
{
        return uc;
    }
    return null;
}

public Usuario obtenerSesion(){
    return obtenerSesion;
}
}

3.4. ControladorAutomovil
public class ControladorAutomovilP extends ControladorAbstractoP<AutomovilP>
implements Runnable {


    private JLabel boton;
    private int x;
    private int y;
    Thread t;

    public ControladorAutomovilP() {
        super();
        //t = new Thread(this);
    }

    public ControladorAutomovilP(JLabel boton, int x, int y) {
        this.boton = boton;
        this.x = x;
        this.y = y;
        t = new Thread(this);
        t.start();
    }

    @Override
    public void run() {
        while (true) {
            this.mover(boton, t, x, y);
        }
    }
}

```


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

3.5. ControladorTicket

```
public class ControladorTicketP extends ControladorAbstractoP<TicketP>{

    public ControladorTicketP() {
        super();
    }

    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);
            }
            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;
        }
    }
}
```

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

```

    }
    if (true) {

    }
    return 0;
}
}

```

4. Funcionamiento de la aplicación


4.1. VntPrincipal

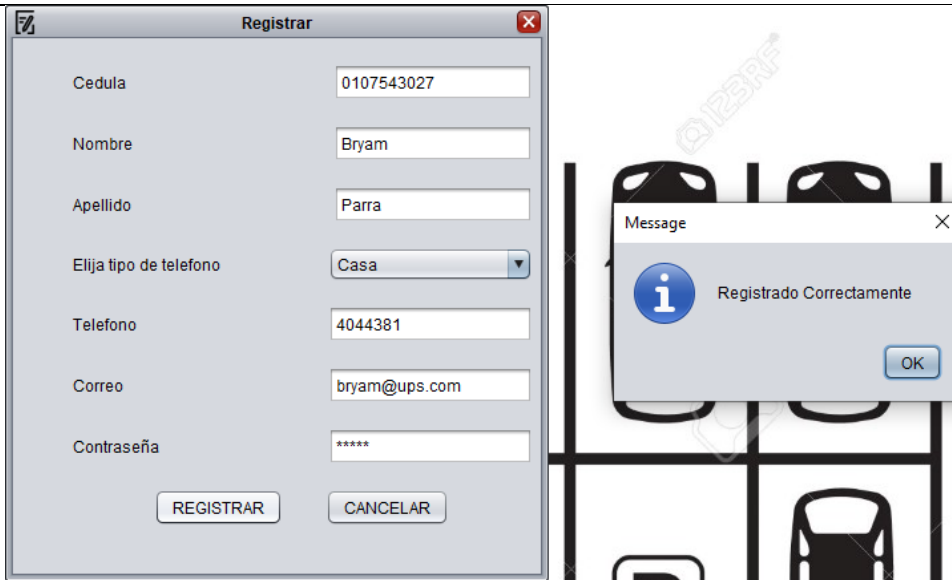
Las ventanas tendrán un menú el cual se desplegara, en el menú Registrar Administrador, se registrara un administrador



4.2. 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.

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



The screenshot shows a 'Registrar' window with the following fields filled out:

- Cedula: 0107543027
- Nombre: Bryam
- Apellido: Parra
- Elija tipo de telefono: Casa
- Telefono: 4044381
- Correo: bryam@ups.com
- Contraseña: *****

Buttons at the bottom: REGISTRAR, CANCELAR. A message box on the right says 'Registrado Correctamente' with an OK button.

4.3. VntRegistrarUsuario

En esta ventana se creará el usuario, validando que se ingresen bien los campos cedula, correo y teléfono, usando reflexión y consultando con queries.




The screenshot shows a 'Registrar' window with the following fields filled out:

- Cedula: 0102680899
- Nombre: eduardo
- Apellido: parra
- Elija tipo de telefono: Movil
- Telefono: 09883987717
- Correo: eduardo@hotmail.com
- Contraseña: *****

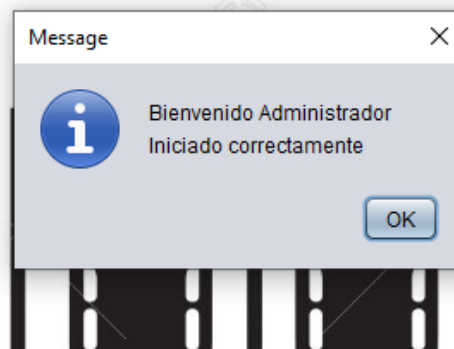
Buttons at the bottom: REGISTRAR, CANCELAR.

4.4. VntIniciarSesion

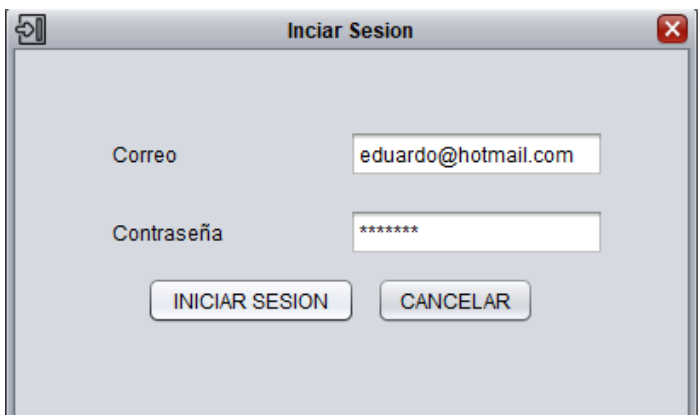
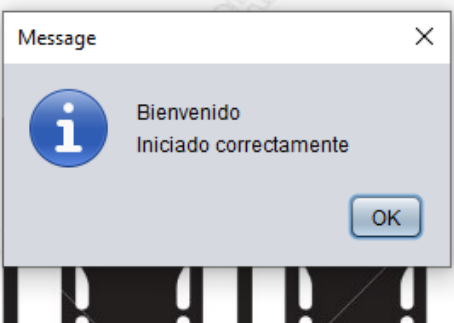
Cuando el administrador inicia sesión en el menú se visualizará la opción de registrar a los usuarios. Cuando el usuario inicia sesión se habilitará un nuevo menú llamado gestión en el que podrá reservar un estacionamiento y generar un ticket.


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

Administrador

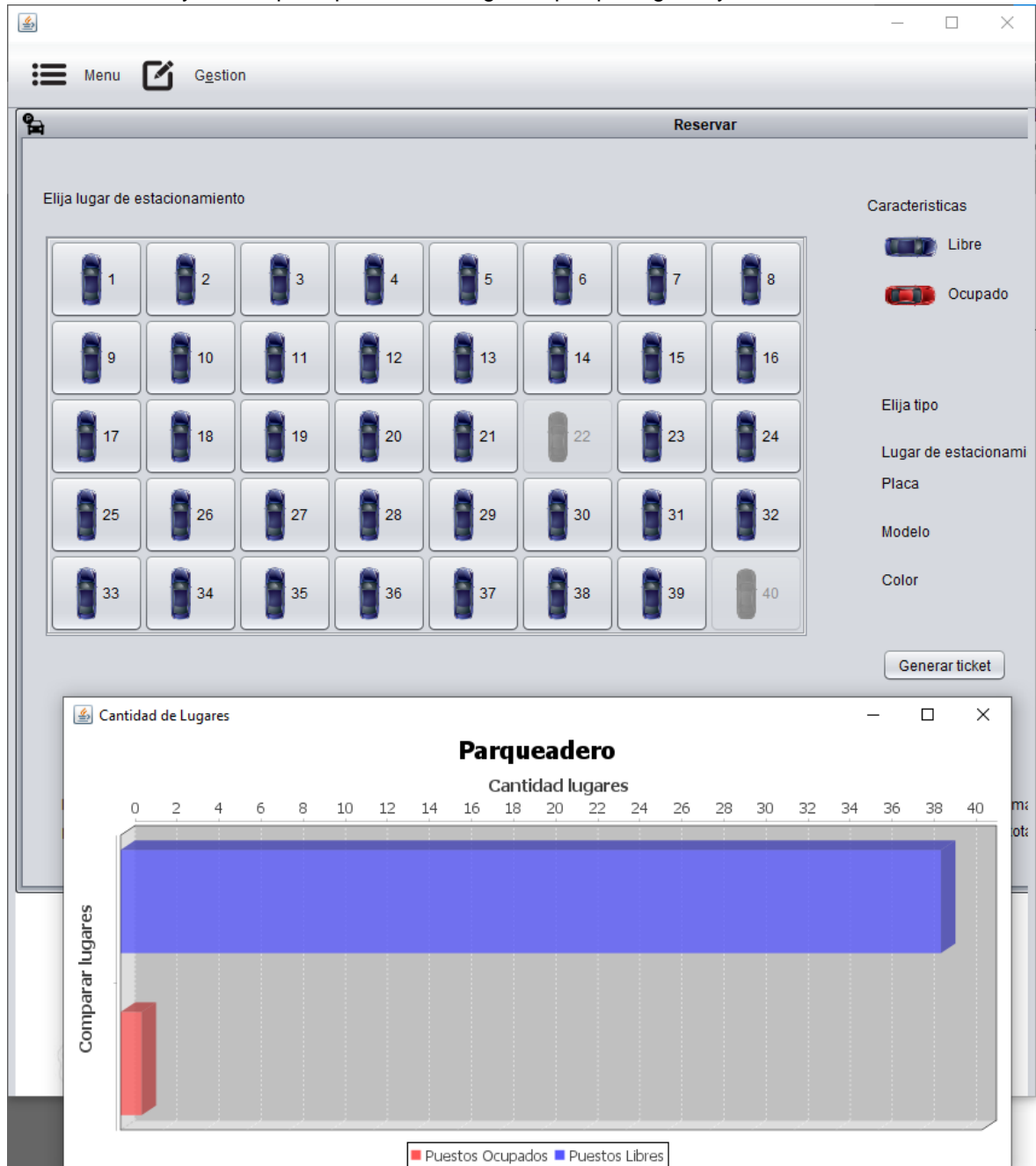
Usuario

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

4.5. VntReservar

Al Iniciar la ventana visualizar un gráfico 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. Al seleccionar el lugar a estacionar, mediante hilos, simulara un vehículo moviéndose hacia el lugar seleccionado, si ya no se pudo presionar el lugar es porque alguien ya lo tiene reservado.



Reservar

Elija lugar de estacionamiento

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Características

Libre

Ocupado

Elija tipo

Lugar de estacionamiento

Placa

Modelo

Color

Generar ticket

Cancelar

Tarifas

Por hora	\$0.50
Por Día	\$10
Por Semana	\$60
Por mes	\$210

--- Elija tipo de contrato ---

13

IMPORTANTE : Si el usuario opto por horas si pasa de las 5h, se le sumara un 10% al valor a pagar, si opto por días al pasar de 5 días se le sumara también un 10%, si opto por la opción por semana al pasar 4 semanas también se le sumara un 10% al total y si opto por mes al pasar 1 mes también se le aumentara el 10% a su total. GRACIAS.

Reservar

Elija lugar de estacionamiento

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Características

Libre

Ocupado

Elija tipo

Lugar de estacionamiento

Placa

Modelo

Color

Generar ticket

Cancelar

Tarifas

Por hora	\$0.50
Por Día	\$10
Por Semana	\$60
Por mes	\$210

--- Elija tipo de contrato ---

13

IMPORTANTE : Si el usuario opto por horas si pasa de las 5h, se le sumara un 10% al valor a pagar, si opto por días al pasar de 5 días se le sumara también un 10%, si opto por la opción por semana al pasar 4 semanas también se le sumara un 10% al total y si opto por mes al pasar 1 mes también se le aumentara el 10% a su total. GRACIAS.

Reservar

Elija lugar de estacionamiento

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Características

Libre

Ocupado

Tarifas

Por hora	\$0.50
Por Día	\$10
Por Semana	\$60
Por mes	\$210

Elija tipo

--- Elija tipo de contrato ---

Lugar de estacionamiento

13

Placa

Modelo

Color

Generar ticket

Cancelar

IMPORTANTE : Si el usuario opto por horas si pasa de las 5h, se le sumara un 10% al valor a pagar, si opto por días al pasar de 5 días se le sumara también un 10%, si opto por la opción por semana al pasar 4 semanas también se le sumara un 10% al total y si opto por mes al pasar 1 mes también se le aumentara el 10% a su total. GRACIAS.

Reservar

Elija lugar de estacionamiento

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Características

Libre

Ocupado

Tarifas

Por hora	\$0.50
Por Día	\$10
Por Semana	\$60
Por mes	\$210

Elija tipo

--- Elija tipo de contrato ---

Lugar de estacionamiento

13

Placa

Modelo

Color

Generar ticket


Cancelar


Reservar

Elija lugar de estacionamiento

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Características

 Libre

 Ocupado

Elija tipo

Lugar de estacionamiento

Placa

Modelo

Color

Generar ticket

Cancelar

Tarifas

Por hora	\$0.50
Por Día	\$10
Por Semana	\$60
Por mes	\$210

Por Dias

13

UBY0335

Nissan

Blanco

IMPORTANTE : Si el usuario opto por horas si pasa de las 5h, se le sumara un 10% al valor a pagar, si opto por días al pasar de 5 días se le sumara también un 10%, si opto por la opción por semana al pasar 4 semanas también se le sumara un 10% al total y si opto por mes al pasar 1 mes también se le aumentara el 10% a su total. GRACIAS.

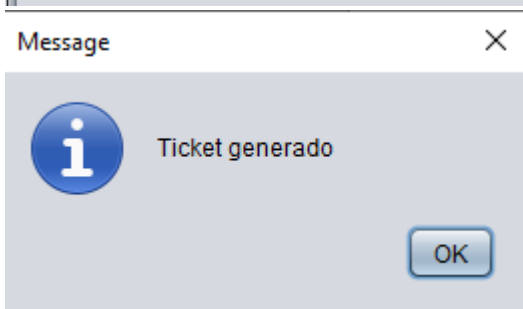
Al hacer clic en generar ticket abrirá una ventana en el que visualizará los datos que tendrá el ticket con los datos del usuario, el auto y del ticket. Al presionar confirmar inmediatamente visualizará un mensaje confirmando el ticket y mostrando el id de ese ticket, que será de utilidad a un futuro


Ticket [X]

Revise si los datos son correctos

ID Ticket	<input type="text"/>
Cedula	<input type="text" value="0102680899"/>
Nombre	<input type="text" value="eduardo"/>
Apellido	<input type="text" value="parra"/>
Telefono	<input type="text" value="0988398771"/>
Numero de estacionamiento	<input type="text" value="13"/>
Placa	<input type="text" value="UBY0335"/>
Modelo	<input type="text" value="Nissan"/>
Color	<input type="text" value="Blanco"/>
Tipo de Contrato	<input type="text" value="Por Dias"/>
Fecha de Ingreso	<input type="text" value="2021-02-07T23:05:41.944293900"/>

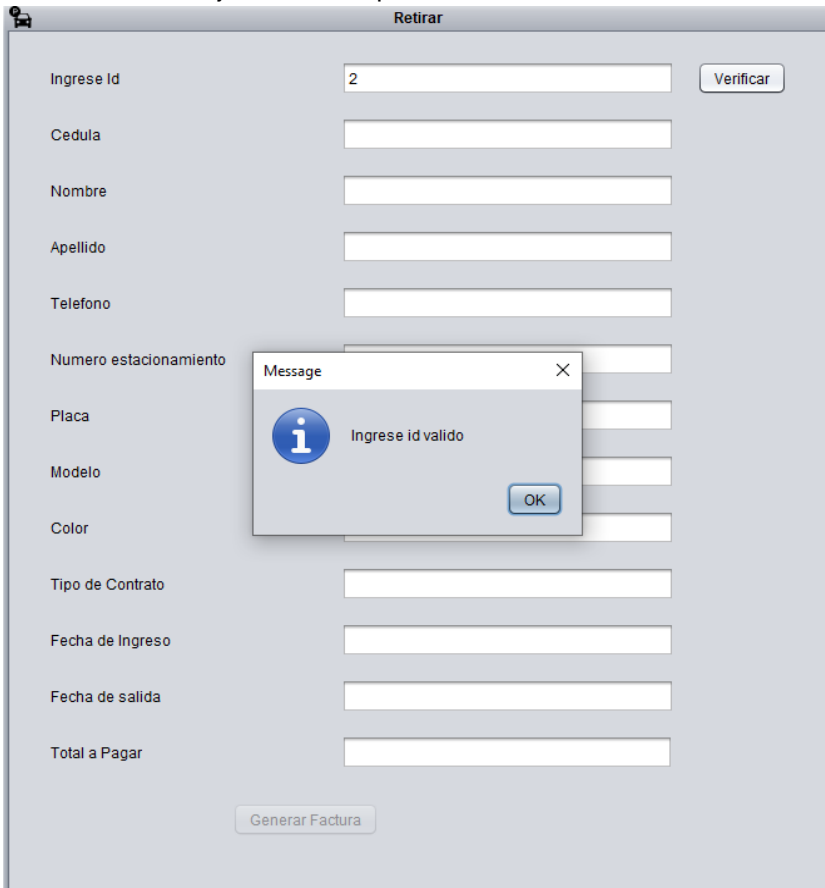
 **Confirmar y Generar**




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

4.6. 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 visualizará mensaje indicando que ese id no existe.




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



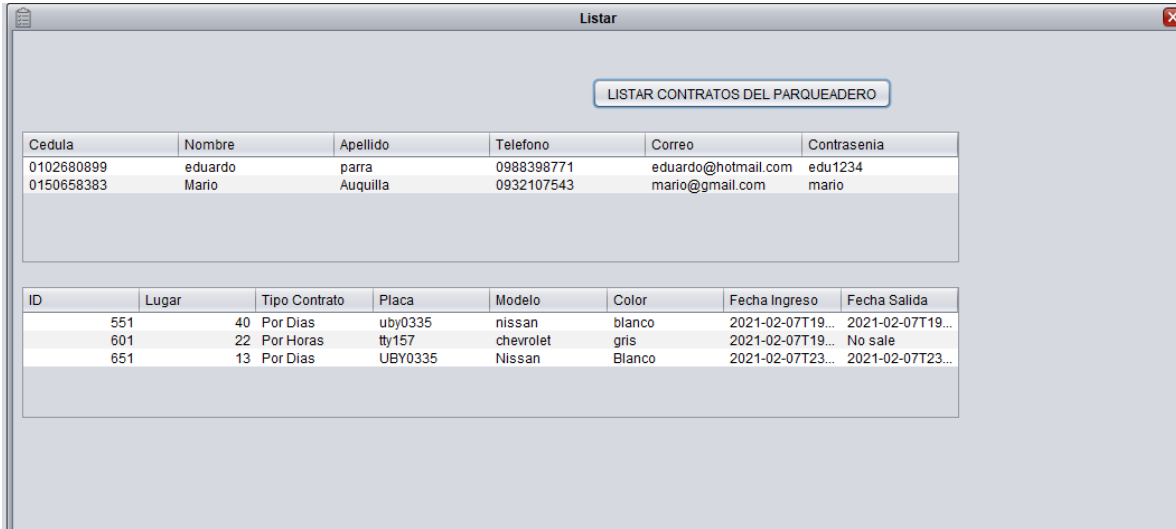
Retirar

Ingrese Id	<input type="text" value="651"/>	<input type="button" value="Verificar"/>
Cedula	<input type="text" value="0102680899"/>	
Nombre	<input type="text" value="eduardo"/>	
Apellido	<input type="text" value="parra"/>	
Telefono	<input type="text" value="0988398771"/>	
Numero estacionamiento	<input type="text" value="13"/>	
Placa	<input type="text" value="UBY0335"/>	
Modelo	<input type="text" value="Nissan"/>	
Color	<input type="text" value="Blanco"/>	
Tipo de Contrato	<input type="text" value="Por Dias"/>	
Fecha de Ingreso	<input type="text" value="2021-02-07T23:05:41.944293900"/>	
Fecha de salida	<input type="text" value="2021-02-07T23:08:07.952152"/>	
Total a Pagar	<input type="text" value="5.0"/>	

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

4.7. VntListarEstacionamiento

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



Cedula	Nombre	Apellido	Telefono	Correo	Contraseña
0102680899	eduardo	parra	0988398771	eduardo@hotmail.com	edu1234
0150658383	Mario	Aquilla	0932107543	mario@gmail.com	mario

ID	Lugar	Tipo Contrato	Placa	Modelo	Color	Fecha Ingreso	Fecha Salida
551	40	Por Dias	uby0335	nissan	blanco	2021-02-07T19...	2021-02-07T19...
601	22	Por Horas	tty157	chevrolet	gris	2021-02-07T19...	No sale
651	13	Por Dias	UBY0335	Nissan	Blanco	2021-02-07T23...	2021-02-07T23...

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.
- Aplica los conocimientos adquiridos de JPA base de datos y de hilos (Thread)

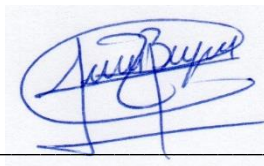
CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos. Los estudiantes implementan soluciones graficas 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: _____