

CARRERA: COMPUTACIÓN/INGENIERÍA DE SISTEMAS

ASIGNATURA: programación aplicada

NRO. Proyecto

1.1

TÍTULO PRÁCTICA: Proyecto integrador Inter ciclo

OBJETIVO ALCANZADO:

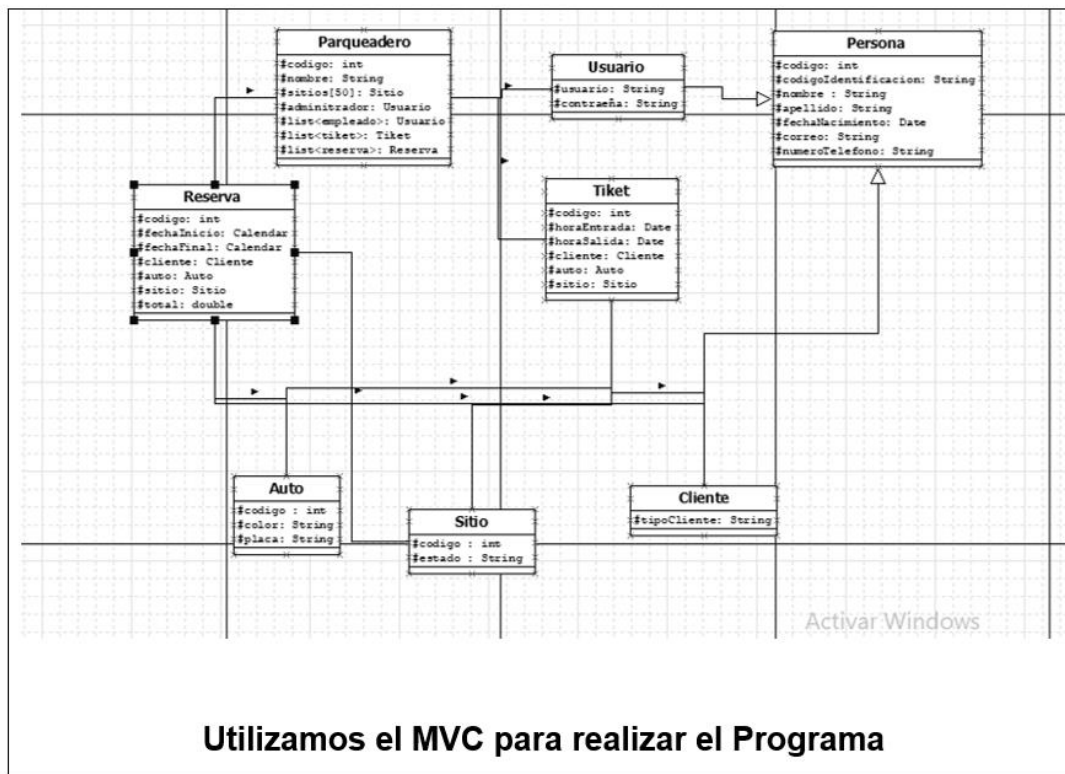
Implementaciones sobre los temas tratados en el primer Inter ciclo de programación aplicada donde están temas considerados como, programación genérica, reflexión expresiones regulares y patrones de diseño. De una manera donde estos temas sean funcionales dentro del entorno Java.

ACTIVIDADES DESARROLLADAS

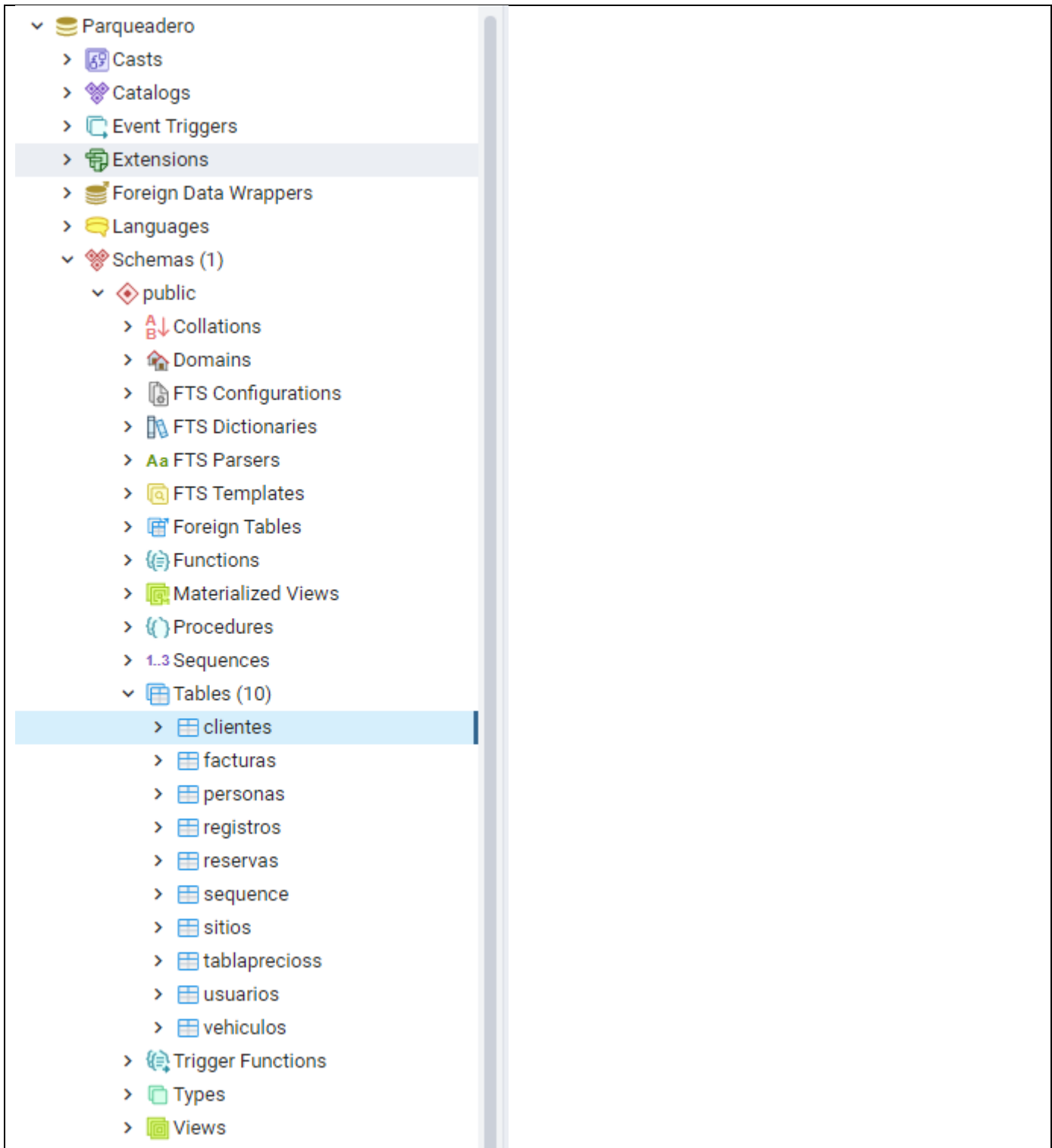
1.Desarrollar un manual para la gestión de parqueaderos.

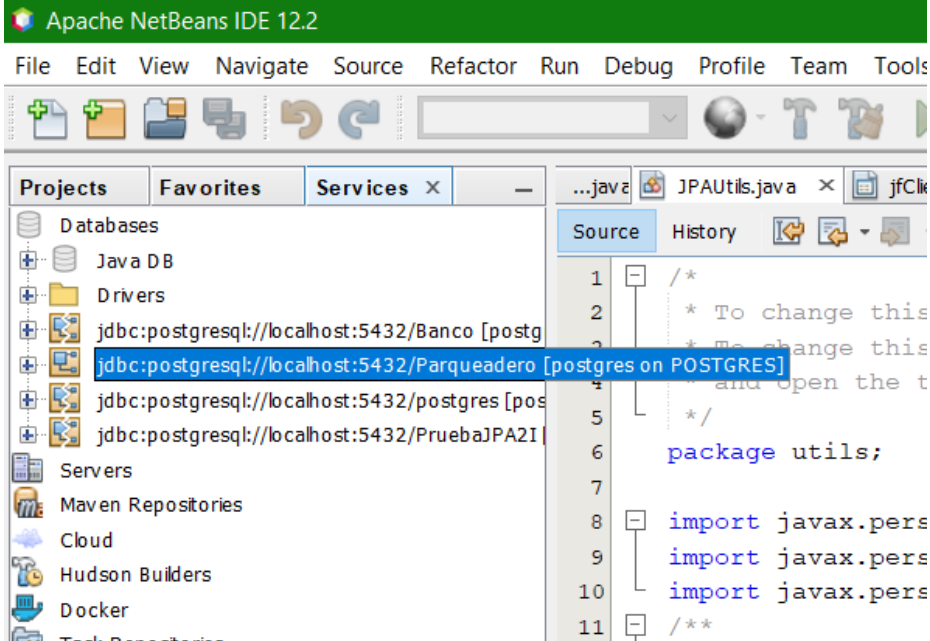
https://prezi.com/p/edit/48aanw_y4gp6/

2.Diagrama Uml



Conexión a base de datos.





Conexiones con java

```
package utils;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 * @author ASUS
 */


public class JPAUtils {

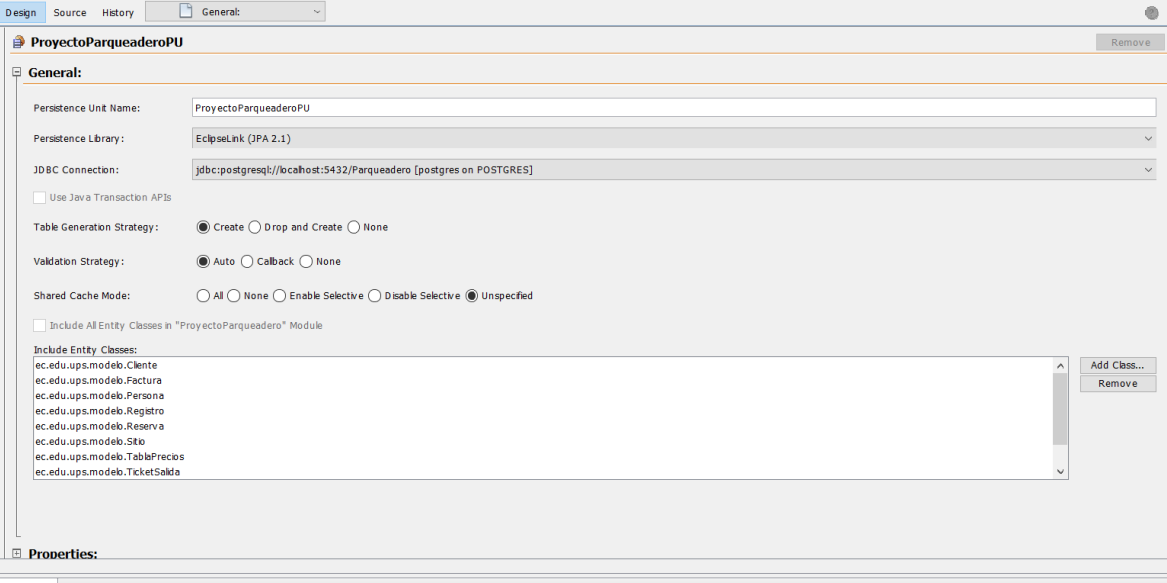
    private static final EntityManagerFactory emf = Persistence.createEntityManagerFactory("ProyectoParqueaderoPU");

    public static EntityManagerFactory getEntityManagerFactory() {
        return emf;
    }

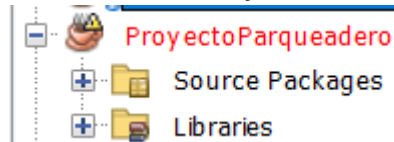
    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }
}
```

Crear el META-INF

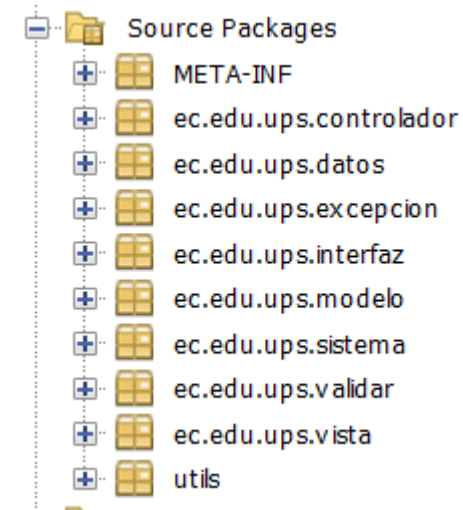
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		




2.Creacion del Proyecto.

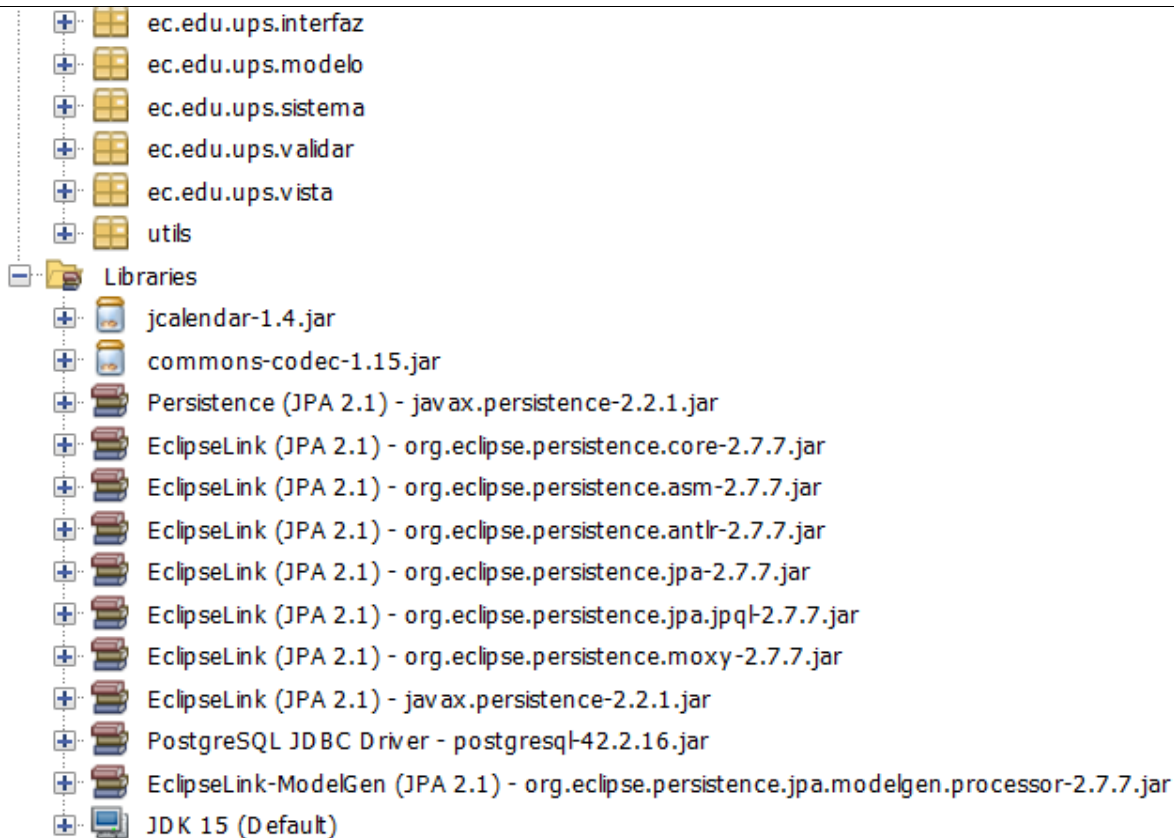


3.Creacion de paquetes a trabajar.



Librerías usadas .

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



A continuación se procede a ilustrar las evidencias del JPA implementadas en el paquete modelo como se puede evidenciar a continuación.

Clase cliente

```
5  */
6  package ec.edu.ups.modelo;
7
8  import java.util.Objects;
9  import javax.persistence.Column;
10 import javax.persistence.Entity;
11 import javax.persistence.NamedQueries;
12 import javax.persistence.NamedQuery;
13 import javax.persistence.Table;
14
15 /**
16  *
17  */
18
19 @Entity
20 @Table(name = "clientes")
21 @NamedQueries({
22     @NamedQuery(name = "cliente.findAll", query = "SELECT c FROM Cliente c")
23 })
24 public class Cliente extends Persona {
25     @Column(name = "tipo_Cliente")
26     private String TipoCliente;
27
28     public Cliente() {
29     }
30
31     public Cliente(String TipoCliente,
32                     int codigo,
33                     String codigoIdentificacion,
34                     String nombre,
35                     String apellido,
36                     String correo,
37                     String numeroTelefono) {
```

Clase factura

```

24  /**
25   *
26   */
27  @Entity
28  @Table(name = "facturas")
29  @NamedQueries({
30      @NamedQuery(name = "Factura.findAll", query = "SELECT f FROM Factura f"),
31      @NamedQuery(name = "Factura.findByCodigo", query = "SELECT f FROM Factura f WHERE f.codigo= :codigo"),})
32  public class Factura implements Serializable {
33
34      @Id
35      @GeneratedValue(strategy = GenerationType.IDENTITY)
36      @Column(name = "codigo")
37      private int codigo;
38      @Column(name = "fecha")
39      @Temporal(javax.persistence.TemporalType.TIME)
40      private Calendar fecha;
41      @OneToOne
42      @JoinColumn(name = "registro_id", nullable = false)
43      private Registro registro;
44      @OneToOne
45      @JoinColumn(name = "reserva_id", nullable = false)
46      private Reserva reserva;
47      private double iva;
48      @Column(name = "total_Pagar")
49      private double totalapagar;
50
51      public Factura() {
52
53

```

Clase persona.

```

- - -
@Entity
@Table(name = "Personas")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
@NamedQueries({
    @NamedQuery(name = "Persona.findAll", query = "SELECT p FROM Persona p"),
    @NamedQuery(name = "Persona.findByCedula", query = "SELECT p FROM Persona p WHERE p.codigoIdentificacion = :cedula"),})
public class Persona implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "codigo")
    private int codigo;
    @Column(name = "cedula", nullable = false, length = 10)
    private String codigoIdentificacion;
    @Column(name = "nombre")
    private String nombre;
    @Column(name = "apellido")
    private String apellido;
    @Column(name = "correo")
    private String correo;
    @Column(name = "numeroTelefono")
    private String numeroTelefono;

    public Persona() {

    }
}

```

Clase registro

```

28  /**
29  @Entity
30  @Table(name = "registros")
31  @NamedQueries({
32      @NamedQuery(name = "Registro.findAll", query = "SELECT s FROM Registro s"),
33      @NamedQuery(name = "Registro.findBycodigo", query = "SELECT s FROM Registro s WHERE s.codigo = :codigo"),})
34  public class Registro implements Serializable {
35
36      @Id
37      @GeneratedValue(strategy = GenerationType.IDENTITY)
38      @Column(name = "codigo")
39      private int codigo;
40      @Column(name = "ingreso")
41      @Temporal(javax.persistence.TemporalType.DATE)
42      private Calendar ingreso;
43      @Column(name = "salida")
44      @Temporal(javax.persistence.TemporalType.DATE)
45      private Calendar salida;
46      @Column(name = "costoparqueo")
47
48      private double costoparqueo;
49      @Column(name = "costoservicios")
50      private float costoservicios;
51      @OneToOne
52      @JoinColumn(name = "cliente_id")
53      private Cliente cliente;
54      @OneToOne
55      @JoinColumn(name = "vehiculo_id")
56      private Vehiculo vehiculo;
57      @OneToOne
58      @JoinColumn(name = "usuario_id")
59      private Usuario usuario;
60  }

```

Clase reserva

```

23  /**
24  *
25  */
26  @Entity
27  @Table(name = "reservas")
28  @NamedQueries({
29      @NamedQuery(name = "Reserva.findAll", query = "SELECT r FROM Reserva r"),
30      @NamedQuery(name = "Reserva.findByCodigo", query = "SELECT r FROM Reserva r WHERE r.codigo= :codigo"),})
31  public class Reserva implements Serializable {
32
33      @Id
34      @GeneratedValue(strategy = GenerationType.IDENTITY)
35      @Column(name = "codigo")
36      private int codigo;
37      @Column(name = "fecha_inicio")
38      @Temporal(javax.persistence.TemporalType.DATE)
39      private Calendar fechaInicio;
40      @Column(name = "fecha_fin")
41      @Temporal(javax.persistence.TemporalType.DATE)
42      private Calendar fechaFin;
43      @OneToOne
44      @JoinColumn(name = "sitio_id")
45      private Sitio sitio;
46      @OneToOne
47      @JoinColumn(name = "cliente_id")
48      private Cliente cliente;
49      @OneToOne
50      @JoinColumn(name = "usuario_id")
51      private Usuario usuario;
52      @Column(name = "total")
53      private double total;

```

Clase sitio.


```
@Entity
@Table(name = "sitios")
@NamedQueries({
    @NamedQuery(name = "Sitio.findAll", query = "SELECT s FROM Sitio s"),
    @NamedQuery(name = "Sitio.findBycodigo", query = "SELECT s FROM Sitio s WHERE s.codigo = :codigo"),})
public class Sitio implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "codigo")
    private int codigo;
    @Column(name = "costoNormal")
    private float costoNormal;
    @Column(name = "costoReserva")
    private float costoReserva;
    @Column(name = "estado")
    private String estado;
    @Column(name = "descripcion")
    private String descripcion;
    @OneToOne
    @JoinColumn(name = "registro_id")
    private Registro registro;
    @OneToOne
    @JoinColumn(name = "reserva_id")
    private Reserva reserva;

    public Sitio() {
```

Clase tabla de precios

```
 */
@Entity
@Table(name = "TablaPrecios")
@NamedQueries({
    @NamedQuery(name = "TablaPrecios.findAll", query = "SELECT r FROM TablaPrecios r"),
    @NamedQuery(name = "TablaPrecios.findByCodigo", query = "SELECT r FROM TablaPrecios r WHERE r.codigo= :codigo"),})
public class TablaPrecios implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "codigo")
    private int codigo;
    @Column(name = "observacion")
    private String observacion;
    @Column(name = "horas")
    private int horas;
    @Column(name = "precio")
    private float precio;

    public TablaPrecios() {

    }

    public TablaPrecios(int codigo, String observacion, float precio, int horas) {
        this.codigo = codigo;
        this.observacion = observacion;
        this.precio = precio;
        this.horas = horas;
    }
}
```

Clase ticket fabrica

```

/**
package ec.edu.ups.modelo;

import ec.edu.ups.interfaz.ITicket;

/**
 *
 */
public class TicketFabrica {

    public ITicket getTipo(String Tipo, String contenido) {
        switch (Tipo) {
            case "ingreso":
                return new TicketIngreso(contenido);
            case "salida":
                return new TicketSalida(contenido);
            case "reserva":
                return new TicketReserva(contenido);
            default:
                return null;
        }
    }
}

```

Clase ticket de ingreso

```

package ec.edu.ups.modelo;

import ec.edu.ups.controlador.ControladorJfTicket;
import ec.edu.ups.interfaz.ITicket;
import ec.edu.ups.vista.jfTicket;

/**
 *
 */
public class TicketIngreso implements ITicket {


    String texto;

    public TicketIngreso(String texto) {
        this.texto = texto;
    }

    @Override
    public void mostrar() {
        jfTicket ticket = new jfTicket();
        ControladorJfTicket controladorJfticket = new ControladorJfTicket("TICKET DE INGRESO", texto, ticket);
        controladorJfticket.iniciar();
    }
}

```

Clase ticket salida

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

```

*/
@Entity
@Table(name = "registros")
@NamedQueries({
    @NamedQuery(name = "TicketSalida.findAll", query = "SELECT s FROM TicketSalida s"),
    @NamedQuery(name = "TicketSalida.findByCodigo", query = "SELECT s FROM TicketSalida s WHERE s.codigo = :codigo"),})
public class TicketSalida implements Serializable, ITicket {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "codigo")
    int codigo;
    @Column(name = "texto")
    String texto;

    public TicketSalida() {
    }

    public TicketSalida(String texto) {
        this.texto = texto;
    }

    @Override
    ...
}

```

Clase usuario

```

@Entity
@Table(name = "Usuarios")
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByCedula", query = "SELECT u FROM Usuario u WHERE u.nombreUsuario = :nombreUsuario"),})
public class Usuario extends Persona {

    @Column(name = "nombreUsuario")
    private String nombreUsuario;
    @Column(name = "contraseña")
    private String contraseña;
    @Column(name = "perfilUsuario")
    private String perfilUsuario;

    public Usuario() {
    }

    public Usuario(String nombreUsuario, String contraseña, int codigo, String codigoIdentificacion, String nombre, String apellido, String correo,
        super(codigo, codigoIdentificacion, nombre, apellido, correo, numeroTelefono);
        this.nombreUsuario = nombreUsuario;
        this.contraseña = DigestUtils.md5Hex(contraseña);//encriptando contraseña
    }

    public String getNombreUsuario() {
        return nombreUsuario;
    }
}

```

Clase usuario

```

@Entity
@Table(name = "vehiculos")
@NamedQueries({
    @NamedQuery(name = "Vehiculo.findAll", query = "SELECT a FROM Vehiculo a"),
    @NamedQuery(name = "Vehiculo.findByCedula", query = "SELECT a FROM Vehiculo a WHERE a.codigo = :codigo"),})
public class Vehiculo implements Serializable {


    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "codigo")
    private int codigo;
    @Column(name = "placa", nullable = false, length = 255)
    private String placa;
    @Column(name = "detalle", nullable = false, length = 255)
    private String detalle;

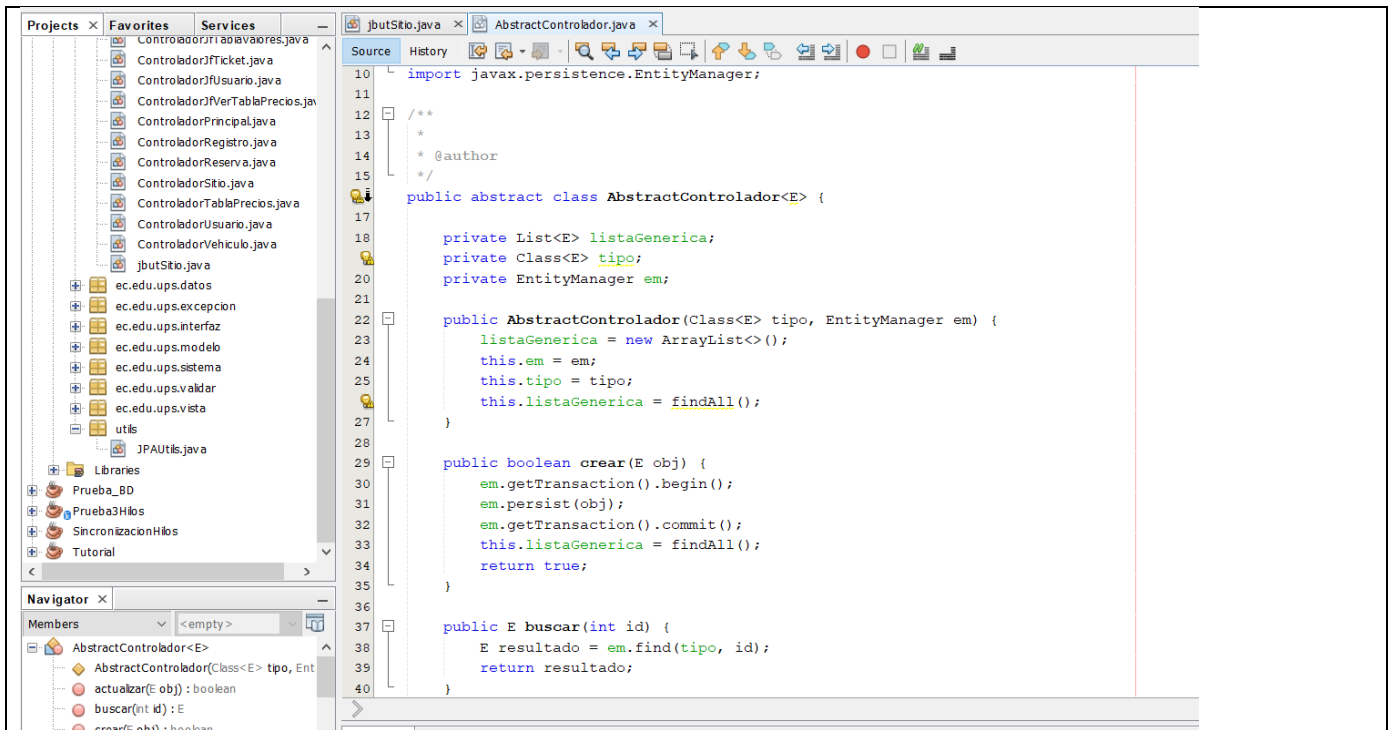
    public Vehiculo() {
    }

    public Vehiculo(int codigo, String placa, String detalle) {
    }
}

```

2.Precentacion de los temas ocupados en el proyecto y en que clases fueron aplicadas.
- Programación genérica.

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



Desarrollo de las clases controlador genérico.

- **Patrones de diseño.**
 - Patrón single ton

Eso en el manejo de archivo.

El singleton es recomendable en clases cuyos objetos solo se deben instanciar una vez para optimizar el uso de memoria además deberá tener un método estático.

```

/**
 *
 * @param <T>
 */
public class ManejadorArchivos<T> {
    private static ManejadorArchivos instancia;
    private ObjectOutputStream salida;//envía los datos a un archivo
    private ObjectInputStream entrada;//lee datos de un archivo
    private List<T> listado;
    private Class<T> tipo;
    //permite al usuario especificar el nombre del archivo
    private ManejadorArchivos(String nombreArchivo){
        try
        {
            salida = new ObjectOutputStream(new FileOutputStream(nombreArchivo));
            entrada = new ObjectInputStream(new FileInputStream(nombreArchivo));
        }catch(IOException ioException){
            System.err.print("Error al abrir el archivo: "+ioException.getMessage());
        }
    }

    public static ManejadorArchivos getInstancia(String nombreArchivo)
    {
        if(instancia == null)
            instancia = new ManejadorArchivos(nombreArchivo);
        return instancia;
    }

    public ObjectOutputStream getSalida() {
        return salida;
    }

    public <T> boolean Escribir()

```

- Patrón Factory.

El Factory se lo usa para delegar la creación de instancias a otro objeto conocido como fábrica. Se lo usó para las validaciones

El Factory tiene por idea delegar la instanciación de clases que tienen un comportamiento similar. Tales como las validaciones por ejemplo reciben un texto y devuelven un resultado True o false.

En este caso esta usado para la generación de los tickets y las validaciones. Para el uso de este patrón fue necesario implementar interfaces para la para la instanciación de diversas clases.

```
package ec.edu.ups.validar;
```

```
import ec.edu.ups.interfaz.IValidar;
```

```
/**  
 *  
 */
```

```
public class ValidarFabrica {  
    public IValidar getTipo(String Tipo)
```

```
{  
    switch(Tipo)  
    {  
        case "email": return new ValidarEmail();  
        case "cedula": return new ValidarCedula();  
        case "entero": return new ValidarEntero();  
        case "texto": return new ValidarTexto();  
        case "float": return new ValidarFloat();  
        case "placa": return new ValidarPlaca();  
        default: return null;  
    }  
}
```

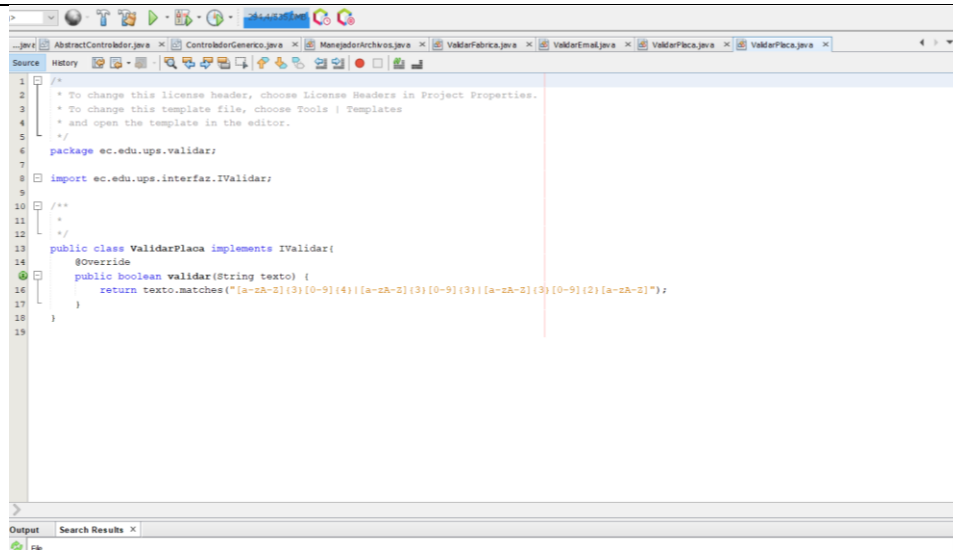
```
}
```

Expresiones regulares.

Desarrollo de la clase validar email usando expresiones regulares

```
public class ValidarEmail implements IValidar{  
    @Override  
    public boolean validar(String texto) {  
        return texto.matches("^[@]+@[^@]+\\. [a-zA-Z]{2,}$");  
    }  
}
```

Desarrollo de la clase Validar placa con expresiones regulares.



```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6
7  package ec.edu.ups.validar;
8
9
10 import ec.edu.ups.interfaz.IValidar;
11
12
13 /**
14  *
15  */
16 public class ValidarPlaca implements IValidar{
17     @Override
18     public boolean validar(String texto) {
19         return texto.matches("[a-zA-Z]{3}[0-9]{4}([a-zA-Z]{3}[0-9]{3}){0-1}([a-zA-Z]{3}[0-9]{2}){a-zA-Z}");
20     }
21 }

```

Desarrollo de la clase validar texto con expresiones regulares.

```

    */
    package ec.edu.ups.validar;

    import ec.edu.ups.interfaz.IValidar;


    /**
     *
     */
    public class ValidarTexto implements IValidar{
        @Override
        public boolean validar(String texto) {
            return texto.matches("[a-zA-Z\\s]+");
        }
    }

```

Además de validar los problemas antes planteados se considero pertinente el controlar una cedula valida, por lo cual se considero pertinente este proceso dentro del paquete de validación como se muestra a continuación.

```
*/  
public class ValidarCedula implements IValidar{  
  
    @Override  
    public boolean validar(String texto) {  
        return validarCedula(texto);  
    }  
    public boolean validarCedula(String cedula) {  
        char digito[] = cedula.toCharArray();  
        int total = 0;  
        for (int i = 0; i < digito.length - 1; i++) {  
            int dato = Integer.parseInt(digito[i] + "");  
            if (i % 2 == 0) {  
                if (dato * 2 > 9) {  
                    dato = (dato * 2) - 9;  
                } else {  
                    dato = dato * 2;  
                }  
            }  
            total += dato;  
        }  
        int ultimo = Integer.parseInt(digito[digito.length - 1] + "");  
        if (total % 10 == 0 && 0 == ultimo) {  
            return true;  
        } else {  
            total = 10 - total % 10;  
            if (total == ultimo) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```

Interfaz para las gestiones

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

Sistema Gestionar Reportes Administrar

CÉDULA.

NOMBRE.

APELLIDO.

CORREO.

NUMERO DE TELÉFONO.

PLACA.

DESCRIPCIÓN.

SITIO.

COSTO NORMAL.

COSTO RESERVA.

FECHA DE INGRESO

FECHA DE SALIDA

COSTO TOTAL.

Sitio después se podrá modificar para el acceso vip y eso.

AutoSave ON Informa Proyecto Interocio

File Home Insert Design Layout References Mailings Review View Help Table Design Layout

Clipboard Paste Copy Format Painter

Sistema Gestionar Reportes Administrar

ID.

CÉDULA.

NOMBRE.

APELLIDO.

CORREO.

NUMERO DE TELÉFONO.

TIPO.

Share Comments

Find Replace Select Dictate Voice Sensitivity Editor

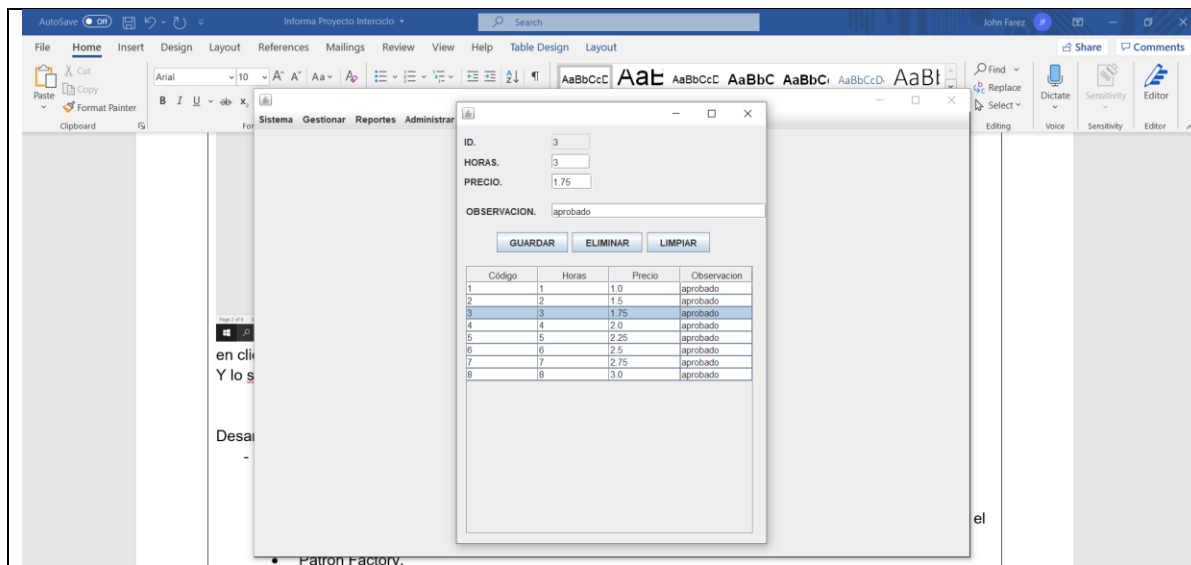
Page 2 of 9 809 words Spanish (Ecuador) Accessibility Investigate

ENG 21:26 14/12/2020

Escribe aquí para buscar

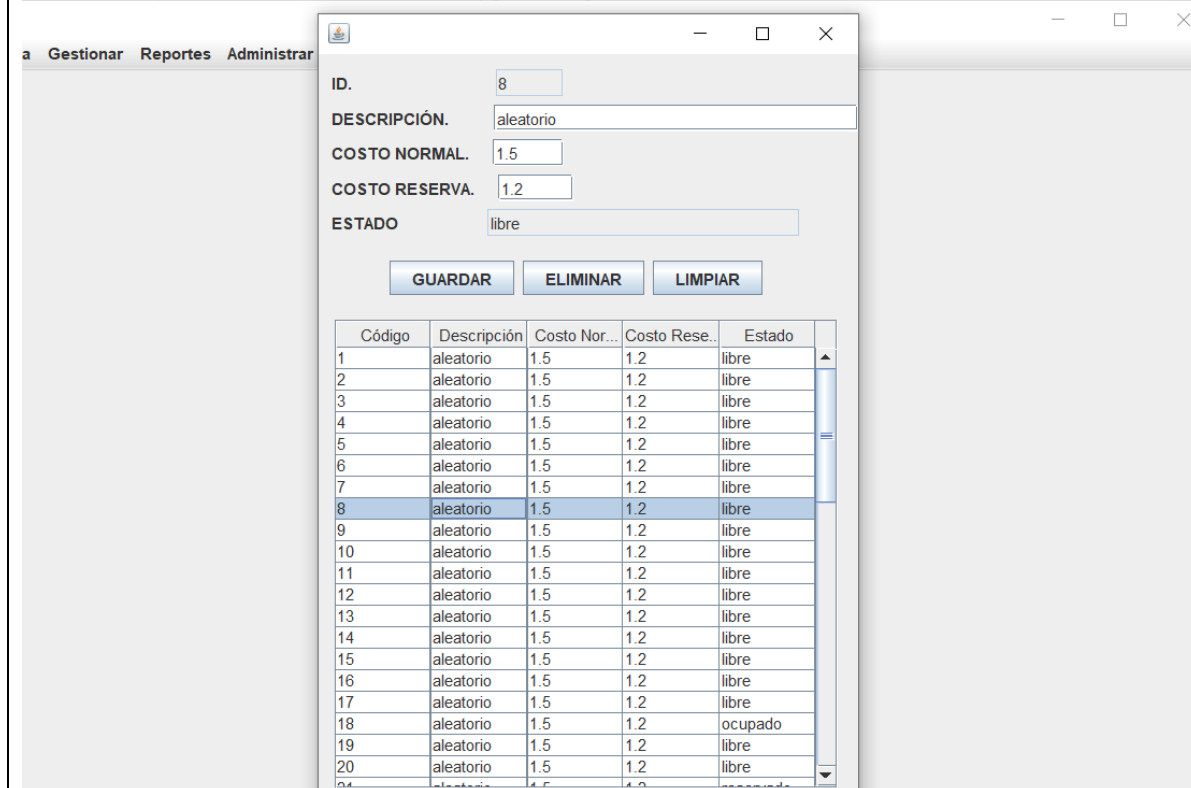
You are screen sharing Stop Share

en clientes se puede hacer el crud para clientes.
Y lo mismo para usuarios.




Código	Horas	Precio	Observacion
1	1	1.0	aprobado
2	2	1.5	aprobado
3	3	1.75	aprobado
4	4	2.0	aprobado
5	5	2.25	aprobado
6	6	2.5	aprobado
7	7	2.75	aprobado
8	8	3.0	aprobado

se puede modificar los precios para precios referenciales para aplicar descuentos a clientes en base a los requerimientos. Es el molde para los costos de cada precio de los parqueaderos y se podrá tener como una base de para poder modificar y una guía para el sistema de cobros.)



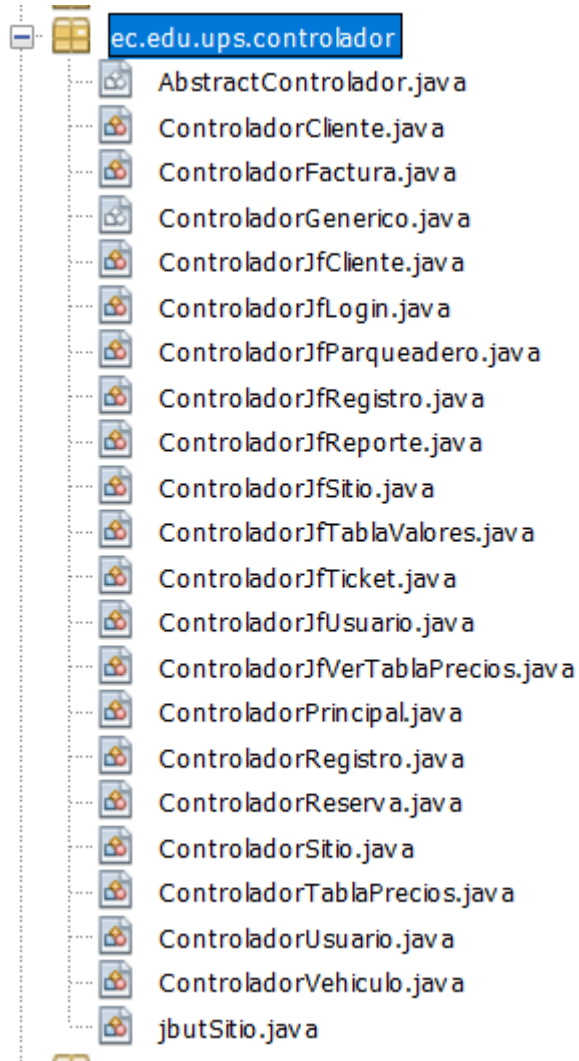
Código	Descripción	Costo Nor...	Costo Rese...	Estado
1	aleatorio	1.5	1.2	libre
2	aleatorio	1.5	1.2	libre
3	aleatorio	1.5	1.2	libre
4	aleatorio	1.5	1.2	libre
5	aleatorio	1.5	1.2	libre
6	aleatorio	1.5	1.2	libre
7	aleatorio	1.5	1.2	libre
8	aleatorio	1.5	1.2	libre
9	aleatorio	1.5	1.2	libre
10	aleatorio	1.5	1.2	libre
11	aleatorio	1.5	1.2	libre
12	aleatorio	1.5	1.2	libre
13	aleatorio	1.5	1.2	libre
14	aleatorio	1.5	1.2	libre
15	aleatorio	1.5	1.2	libre
16	aleatorio	1.5	1.2	libre
17	aleatorio	1.5	1.2	libre
18	aleatorio	1.5	1.2	ocupado
19	aleatorio	1.5	1.2	libre
20	aleatorio	1.5	1.2	libre

si es un cliente frecuente se podrá aplicar los descuentos correspondientes de acuerdo con los requerimientos


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

- Paquete controlador

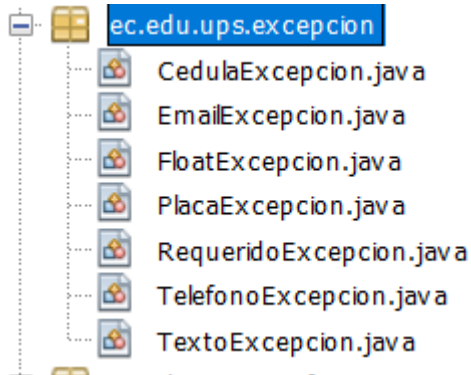
Dentro el paquete controlador estará la funcionalidad y la aplicabilidad de la programación genérica además de el patrón singleton tendiendo así 20 clases normales y 1 abstracta que se encargara de ser un molde para algunas clases y las otras estarán para controlar la interfaz con el usuario teniendo así las clases.



- Paquete excepción

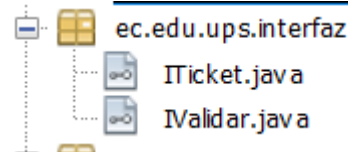
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Dentro del paquete excepción controlaremos en suma mayoría todo lo que tenga que ver con errores del sistema manejables o no manejables es decir el sistema estará en la capacidad de enviar mensajes cuando el usuario haya echo algo que no debía y el sistema podrá orientarlo a buscar otra nueva forma de poder seguir avanzando en el manejo del sistema de parqueadero.



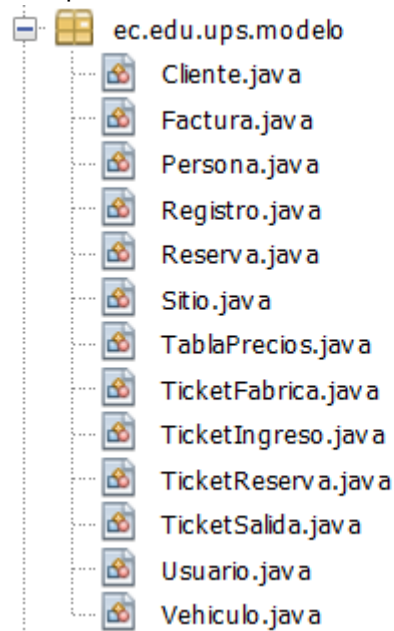
- Paquete interfaz.

Esta clase será como un molde para las otras clase ya que existen muchas clases que necesitan validar y mostrar datos ya sea que estén guardados o que hayan sido creados por el usuario.




- Paquet model.

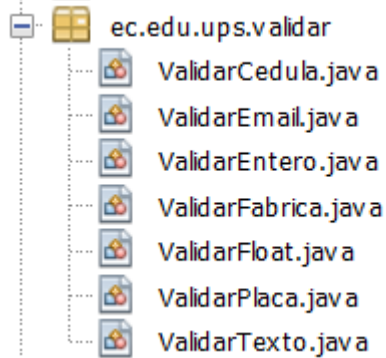
Este paquete será el encargado de manejar en su mayoría los atributos o las características que tendrá el sistema de este paquete se registrarán las características y datos que el parqueadero tendrá y estarán en la capacidad de poder almacenar y modificar, establecer y mostrar datos a otras clases siempre y cuando las implementaciones sean las correctas.



- Paquete validar

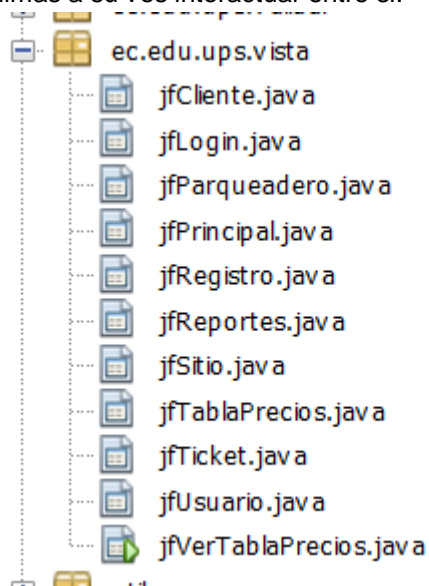
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


En este paquete el usuario deberá regirse en su totalidad la restricciones del sistema informática tales como cédulas validas, correo fiables, contraseñas y datos del vehículo. Ya que esta clase implementa expresiones regulares y así el usuario no podrá escribir algo incoherente dentro de una área de texto que le pida algo en concreto.



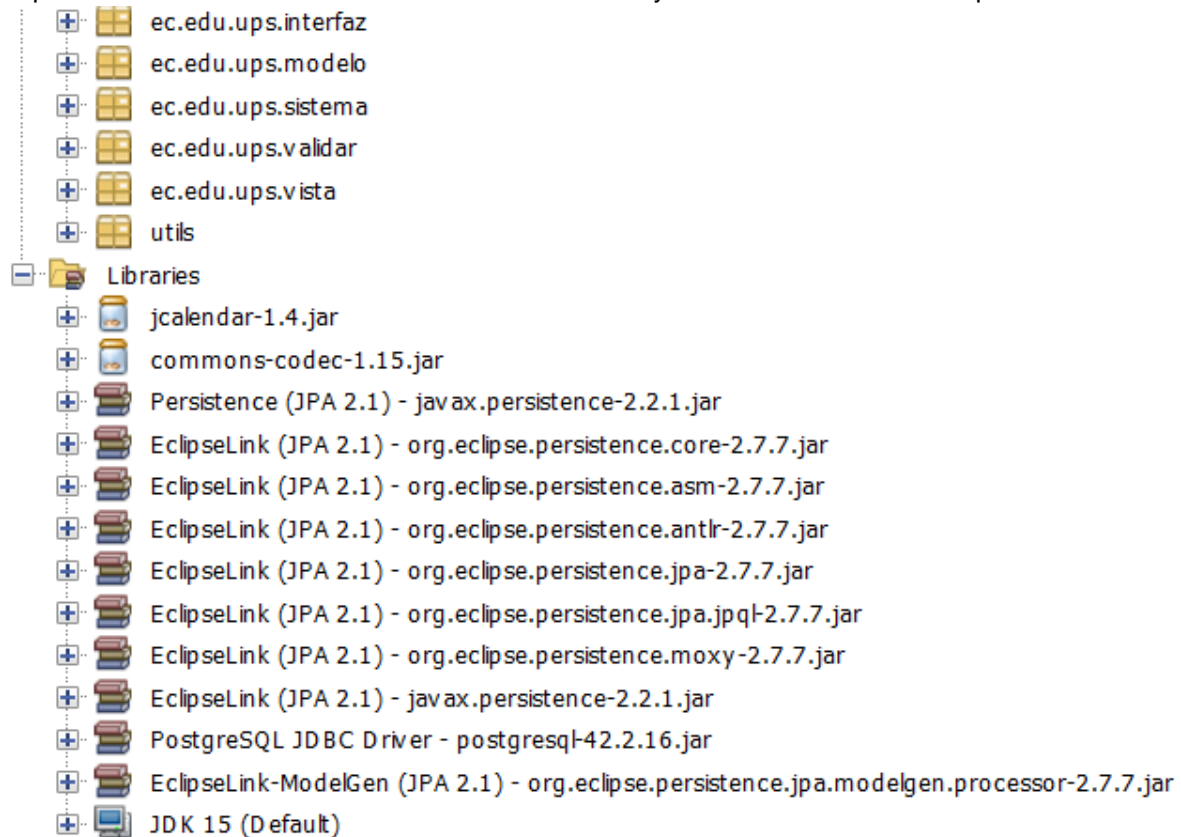
- Paquete vista.

Este paquete es la encargada de el diseño de las ventanas para poder interactuar con el usuario y las mimas a su ves interactuar entre si.



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

Implantación de nuevas librerías tales como Calendar y md5 además de librerías para la base de datos.




El calendar nos sirve para ubicarnos Enel tiempo actual es decir fecha y hora para poder hacer requerimientos funcionales del sistema y el md5 que se uso para la encriptación y desencriptación de el correo y la contraseña.

RESULTADO(S) OBTENIDO(S):

- Desarrollar de una manera adecuada la implantación de temas introductorios a la programación aplicada dentro de java además de funcionales y algoritmos donde se puedan reutilizar para proyectos posteriores dentro de un mismo entorno.
- Investigación acerca de nuevas herramientas para la facilidad de presentaciones tales como son Web 2.0 o presi, que de cierta manera ayudan mucho a la presentación de el trabajo realizado para poder llegar ala finalización del proyecto Inter ciclo.
- Implementación de JPA, postgresql y practica de una conexión eficaz con una base de datos.

CONCLUSIONES:

- El estudiante desarrollo de una manera fiable y entendible la interacción con el usuario además de que se plantea a manera de manual una herramienta web para que el usuario pueda manejar el sistema de una mejor manera y correcta.
- Aplicabilidad de temas propuestos en clases de una manera correcta eh ingeniosa.
- Aplicación de JPA de una manera pertinente.

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

RECOMENDACIONES:

- Desarrollar el Proyecto con mucha anticipación para poder terminar y entender de una mejor manera el funcionamiento de este para así poder lograr el objetivo planteado y poder ir más allá.
- Considerar el hecho de buscar ayuda del profesor para poder solventar cualquier duda que se presente en el momento de desarrollar el proyecto y de esta manera poder cumplir con todos los temas tratados en la rúbrica del trabajo.
- Revisar mas el contenido de la teoría y desarrollar mas ejercicios para poder llevar a cabo un manejo correcto de hilos dentro de java además del JPA

Nombre de estudiante: John Xavier Farez Villa.

Firma de estudiante:

