

Universidad Del Cauca

Facultad De Ingeniería Electrónica Y Telecomunicaciones



Universidad
del Cauca

Programa Ingeniería De Sistemas

Laboratorio Ingeniería de Software

Adrian Felipe Burbano Narvaez

Yerson Argote

Breiner Mamian Jiménez

Popayán
Cauca
27/02/2020

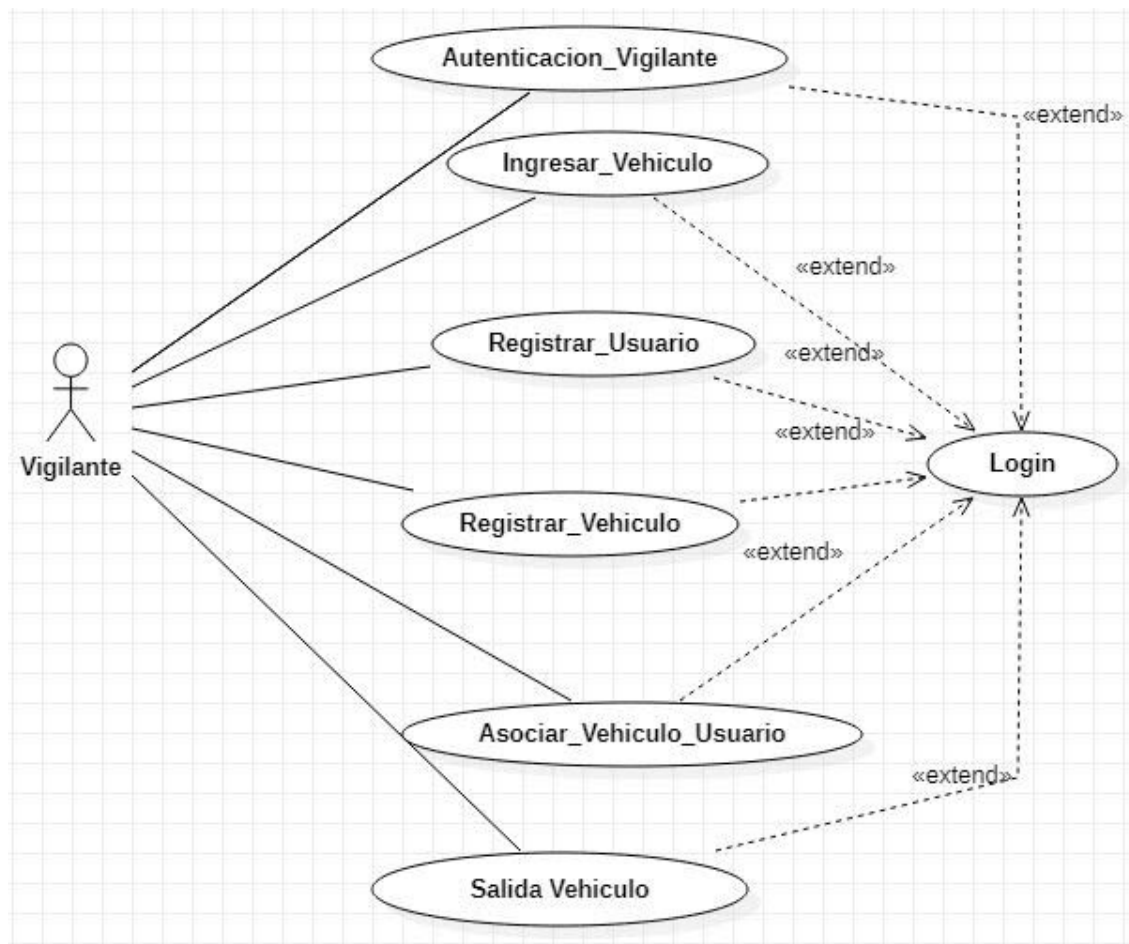
Requisitos para la **Segunda Entrega.**:

Número	Descripción
1	Iniciar sesión para el Vigilante.
2	Ingreso vehículo.
3	Mostrar el mapa de Vehiculos.
4	Asignar un puesto al vehículo.
5	Salida del vehículo.
7	Registrar Datos Conductor.
8	Registrar vehículo y asignarlo a un conductor.
9	Asociar un vehículo a una persona.

Total Número Requisitos **14.**

Avance total en Porcentaje: **57.15%**

Casos de Uso Parqueadero Tulcán.



Caso de uso Número 1.

Nombre	Autenticar Vigilante
Actor	Vigilante.
Precondición	El vigilante debe estar registrado previamente.
Descripción	El vigilante inicia sesión mediante un usuario(login) y contraseña(password).
Flujo Principal	<ol style="list-style-type: none">1. El sistema muestra interfaz de login para Vigilante.2. Vigilante escribe su nombre de usuario.3. Vigilante escribe su contraseña.4. El sistema Valida Nombre de usuario5. El sistema Valida contraseña.6. El sistema muestra interfaz principal..7. fin del caso de uso.
Sub Flujo	<p>4.1 El nombre de usuario no es válido:</p> <p>4.1.1 Mostrar Mensaje de error.</p> <p>4.1.2 regresa al paso 2.</p> <p>5.1 La contraseña no es válida:</p> <p>5.1.1 Mostrar mensaje de error.</p> <p>5.1.2 regresa al paso 3.</p>

Caso de uso número 2.

Nombre	Ingresar Vehículo
Actor	Vigilante.
Precondición	<ol style="list-style-type: none">1. El vigilante debe iniciar sesión.2. El usuario debe estar registrado en el sistema.3. El vehículo debe estar registrado en el sistema.4. El vehículo debe de estar asociado a un usuario.
Descripción	El vigilante ingresa datos del propietario del vehículo y el sistema asigna puesto en el parqueadero.
Flujo Principal	<ol style="list-style-type: none">1. El vigilante recibe el documento del usuario.2. El vigilante ingresa el número de identificación al sistema.3. El sistema verifica número en la base de datos central .4. El sistema muestra los vehículos asociados al usuario.5. El vigilante selecciona el vehículo correspondiente.6. El vigilante elige la opción ingresar.7. El sistema asigna puesto en el parqueadero.8. El sistema muestra ventana de verificación.9. El vigilante presiona aceptar.10. El sistema actualiza el mapa parqueadero.11. Fin del caso de uso
Sub Flujo	<p>3.1 El número de usuario no existe:</p>

	3.1.1 Mostrar Mensaje de error. 3.1.2 regresa al paso 1. 4.1 No tiene vehículos asociados: 4.1.1 Mostrar mensaje de error. 4.1.2 El sistema no despliega vehículos. 4.1.3 regresa al paso 2. 7.1 El parqueadero está lleno: 7.1.1 El sistema muestra mensaje de notificación. 7.1.1 regresa al paso 2
--	---

Caso de uso número 3.

Nombre	Registrar Usuario
Actor	Vigilante.
Precondición	El vigilante debe iniciar sesión.
Descripción	El vigilante ingresa los datos del nuevo usuario al sistema para registrarlo.
Flujo Principal	<ol style="list-style-type: none"> 1. El Vigilante presiona la pestaña gestión usuario. 2. Vigilante ingresa datos del nuevo usuario. 3. Vigilante selecciona tipo de rol. 4. El vigilante presiona el botón registrar. 5. El Sistema registra al usuario. 6. El sistema despliega mensaje de notificación. 7. fin del caso de uso.
Sub Flujo	2.1 El número de identificación ya están en el sistema : 2.1.1 Mostrar Mensaje de error. 2.1.2 regresa a la interfaz principal.

Caso de uso número 4.

Nombre	Registrar Vehículo
Actor	Vigilante.
Precondición	El vigilante debe iniciar sesión.
Descripción	El vigilante ingresa los datos del nuevo Vehículo al sistema para registrarlo.
Flujo Principal	<ol style="list-style-type: none"> 1. El Vigilante presiona la pestaña gestión Vehiculo. 2. El vigilante ingresa la marca,placa y tipo del nuevo vehículo. 3. El vigilante presiona el botón registrar. 4. El sistema registra el Vehículo. 5. El Sistema despliega mensaje notificación. 6. fin del caso de uso.
Sub Flujo	2.1 El número de placa ya está registrada en el sistema :

	2.1.1 Mostrar Mensaje de error. 2.1.2 regresa a la interfaz principal.
--	---

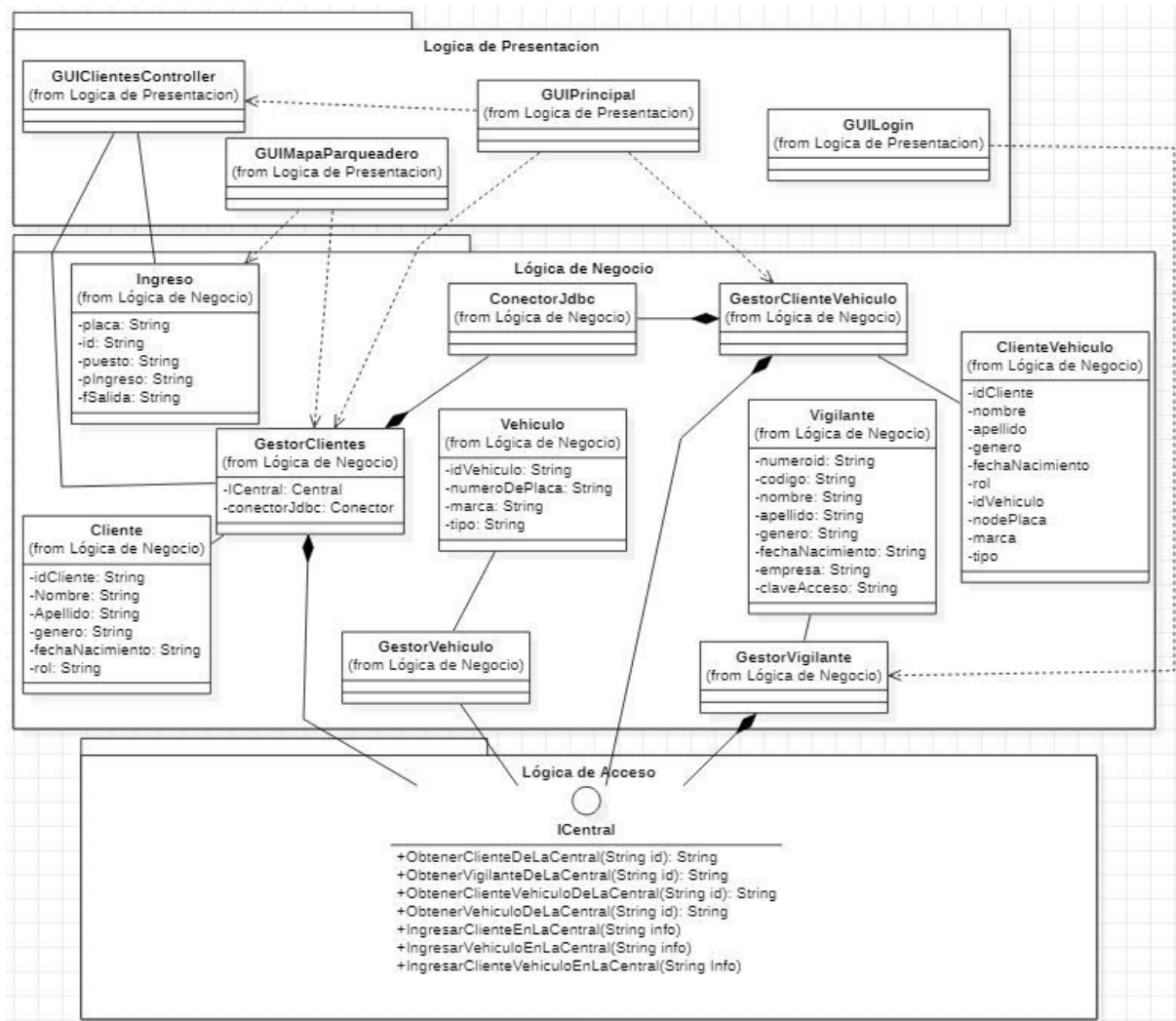
Caso de uso número 5.

Nombre	Asociar Vehículo Usuario
Actor	Vigilante.
Precondición	1. El vigilante debe iniciar sesión. 2. El usuario debe de estar registrado. 3. El vehículo debe de estar registrado.
Descripción	El vigilante ingresa la identificación del usuario y el número de placa del vehículo para asociar .
Flujo Principal	1. El Vigilante presiona la pestaña gestión Asociación. 2. El vigilante ingresa el número de identificación del usuario. 3. El vigilante ingresa la placa del vehículo. 4. El vigilante presiona el botón Realizar Asociación. 5. El Sistema despliega mensaje notificación. 6. fin del caso de uso.
Sub Flujo	2.1 El número de identificación no está registrado en el sistema : 2.1.1 Mostrar Mensaje de error. 2.1.2 regresa al paso número 1. 3.1 La placa ingresada no está registrada en el sistema: 3.1.1 Mostrar mensaje de error. 3.1.2 regresa al paso 1.

Caso de uso número 6.

Nombre	Salida Vehículo
Actor	Vigilante.
Precondición	1. El vigilante debe iniciar sesión. 2. El vehículo debe ingresar previamente.
Descripción	El vigilante ingresa el número de identificación del usuario, elige el respectivo carro y presiona el botón retirar.
Flujo Principal	1. El vigilante ingresa el número de identificación 2. El vigilante selecciona el vehículo. 3. El vigilante presiona el botón retirar. 4. El sistema retira el vehículo del mapa. 5. El sistema actualiza la vista del mapa 6. El Sistema despliega mensaje notificación. 7. fin del caso de uso.
Sub Flujo	3.1 El vehículo seleccionado no ha ingresado previamente : 3.1.1 Mostrar Mensaje de error. 3.1.2 regresa a la interfaz principal.

Diagrama de Clase:



Patrones de Diseño Usados.

1) Patrón Singleton:

Aquí se observa el patrón Singleton en el método estático `getConnectorJdbc` de la clase `ConectorJdbc` donde si el atributo estático `connector` es nulo, se inicializa y se retorna, y si no es nulo simplemente se retorna.

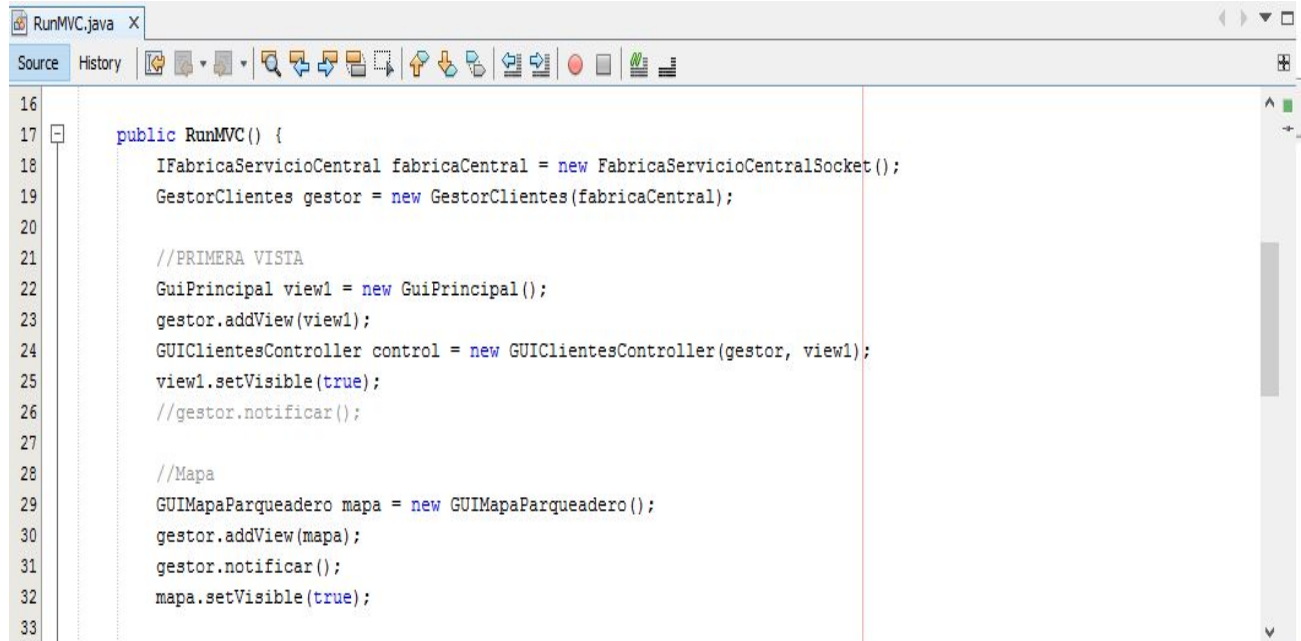
```

SQL 2 [Central] x ConectorJdbc.java x ConectorJdbc.java x
Source History
25     private final String PASSWORD = "123";
26
27     private ConectorJdbc() {
28     }
29
30
31     public static ConectorJdbc getConnectorJdbc() {
32         if (connector == null) {
33             connector = new ConectorJdbc();
34         }
35         return connector;
36     }
37
38     public void conectarse() throws ClassNotFoundException, SQLException {
39         Class.forName("org.hsqldb.jdbcDriver");
40         cn = DriverManager.getConnection(URL, USER, PASSWORD);
41     }
42
43     /**

```

2) Patrón Observer:

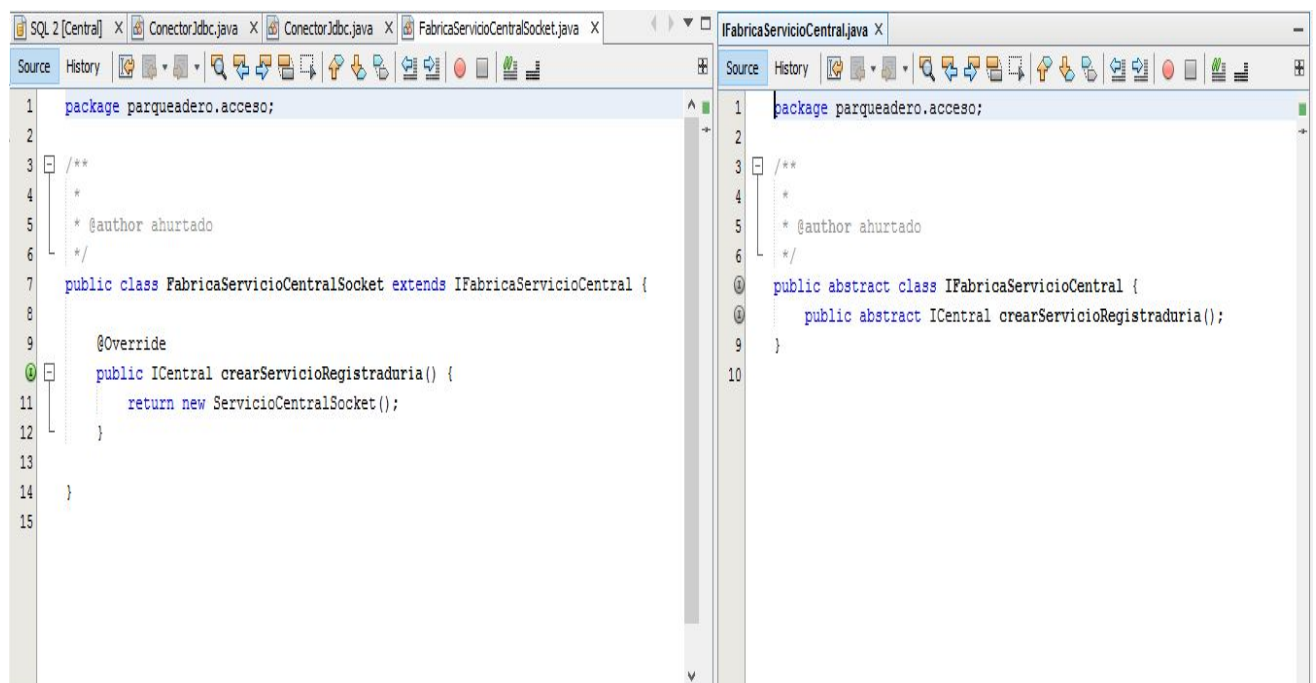
En la siguiente imagen se evidencia la aplicación del patrón observer, donde se crea el modelo (GestorClientes), se le agregan las vistas (GuiPrincipal, GUIMapaParqueadero) y se crea el controlador que comunica el GestorClientes y GuiPrincipal, y el controlador se encarga del manejo de los eventos “INGRESAR” y “RETIRAR” vehículo.



```
16
17 public RunMVC() {
18     IFabricaServicioCentral fabricaCentral = new FabricaServicioCentralSocket();
19     GestorClientes gestor = new GestorClientes(fabricaCentral);
20
21     //PRIMERA VISTA
22     GuiPrincipal view1 = new GuiPrincipal();
23     gestor.addView(view1);
24     GUIClientesController control = new GUIClientesController(gestor, view1);
25     view1.setVisible(true);
26     //gestor.notificar();
27
28     //Mapa
29     GUIMapaParqueadero mapa = new GUIMapaParqueadero();
30     gestor.addView(mapa);
31     gestor.notificar();
32     mapa.setVisible(true);
33 }
```

3) Patrón Factory:

Este patrón es aplicado en la construcción del ServicioCentralSocket que hace la comunicación con Servidor Central.



```
1 package parqueadero.acceso;
2
3 /**
4  *
5  * @author ahurtado
6  */
7 public class FabricaServicioCentralSocket extends IFabricaServicioCentral {
8
9     @Override
10     public ICentral crearResultaduria() {
11         return new ServicioCentralSocket();
12     }
13
14 }
15
```

```
1 package parqueadero.acceso;
2
3 /**
4  *
5  * @author ahurtado
6  */
7 public abstract class IFabricaServicioCentral {
8     public abstract ICentral crearResultaduria();
9 }
10
```

Muestra Aplicativo.



Interfaz Principal.

