TP FINAL

Taller de

Programación I

Integrantes:

* Jose María Gutierrez Paredes
* Tomás Civiero
* Franco Vallone

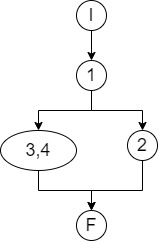
**Test de Caja Blanca**

**Clase: Empresa**

**Método: agregarChofer**

public void agregarChofer(Chofer chofer) throws ChoferRepetidoException {  
 if (this.choferes.containsValue(chofer)) { **#1**  
 throw new ChoferRepetidoException(chofer.getDni(), (Chofer)this.choferes.get(chofer.getDni())); **#2**  
 } else {  
 this.choferes.put(chofer.getDni(), chofer); **#3**  
 this.choferesDesocupados.add(chofer); **#4**  
 }  
}

**Grafo de Flujo**



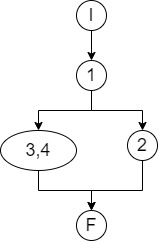
**Caminos B de Flujo**

**Clase: Empresa**

**Método: agregarCliente**

public void agregarCliente(String usuario, String pass, String nombreReal) throws UsuarioYaExisteException {  
 if (this.clientes.containsKey(usuario)) { **#1**  
 throw new UsuarioYaExisteException(usuario); **#2**  
 } else {  
 Cliente cliente = new Cliente(usuario, pass, nombreReal); **#3**  
 this.clientes.put(cliente.getNombreUsuario(), cliente); **#4**  
 }  
}

**Grafo de Flujo**

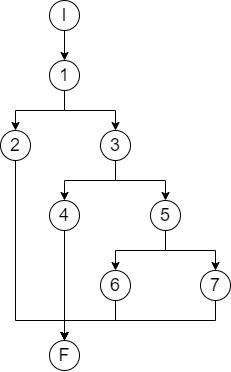


**Clase: Empresa**

**Método: agregarPedido**

public void agregarPedido(Pedido pedido) throws SinVehiculoParaPedidoException, ClienteNoExisteException, ClienteConViajePendienteException, ClienteConPedidoPendienteException {  
 if (!this.clientes.containsValue(pedido.getCliente())) { **#1**  
 throw new ClienteNoExisteException(); **#2**  
 } else if (this.pedidos.containsKey(pedido.getCliente())) { **#3**  
 throw new ClienteConPedidoPendienteException(); **#4**  
 } else if (this.validarPedido(pedido)) { **#5**  
 this.pedidos.put(pedido.getCliente(), pedido); **#6**  
 } else {  
 throw new SinVehiculoParaPedidoException(pedido); **#7**  
 }  
}

**Grafo de Flujo**



**Clase: Empresa**

**Método: agregarVehiculo**

public void agregarVehiculo(Vehiculo vehiculo) throws VehiculoRepetidoException {  
 if (this.vehiculos.containsKey(vehiculo.getPatente())) { **#1**  
 throw new VehiculoRepetidoException(vehiculo.getPatente(), (Vehiculo)null); **#2**  
 } else {  
 this.vehiculos.put(vehiculo.getPatente(), vehiculo); **#3**  
 this.vehiculosDesocupados.add(vehiculo); **#4**  
 }  
}

**Grafo de Flujo**

A diagram of a number and two circles

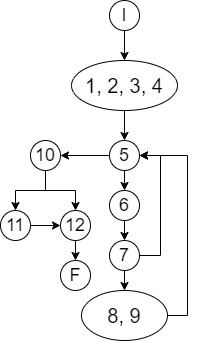
Description automatically generated

**Clase: Empresa**

**Método: calificacionDeChofer**

public double calificacionDeChofer(Chofer chofer) throws SinViajesException {  
 double r = 0.0; **#1**  
 double suma = 0.0; **#2**  
 int cant = 0; **#3**  
 Iterator<Viaje> it = this.viajesTerminados.iterator(); **#4**  
  
 while(it.hasNext()) { **#5**  
 Viaje v = (Viaje)it.next(); **#6**  
 if (v.getChofer() == chofer) { **#7**  
 suma = (double)v.getCalificacion(); **#8**  
 ++cant; **#9**  
 }   
 }  
  
 if (cant > 0) { **#10**  
 r = suma / (double)cant; **#11**  
 }  
  
 return r; **#12**  
}

**Grafo de Flujo**

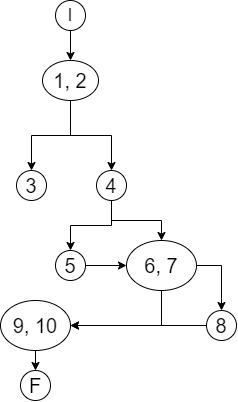


**Clase: Empresa**

**Método: login**

public Usuario login(String usserName, String pass) throws UsuarioNoExisteException, PasswordErroneaException {  
 Usuario u = null; **#1**  
 if (usserName.equals("admin") && pass.equals("admin")) { **#2**  
 u = Administrador.getInstance(); **#3**  
 } else {  
 if (!this.clientes.containsKey(usserName)) { **#4**  
 throw new UsuarioNoExisteException(usserName); **#5**  
 }  
  
 u = (Usuario)this.clientes.get(usserName); **#6**  
 if (!((Usuario)u).getPass().equals(pass)) { **#7**  
 throw new PasswordErroneaException(usserName, pass); **#8**  
 }  
 }  
  
 this.usuarioLogeado = (Usuario)u; **#9**  
 return (Usuario)u; **#10**  
}

**Grafo de Flujo**

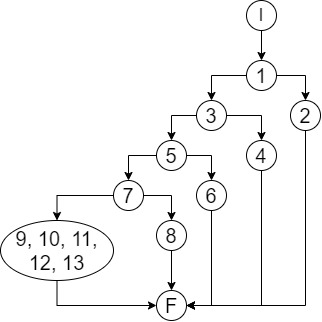


**Clase: Empresa**

**Método: crearViaje**

public void crearViaje(Pedido pedido, Chofer chofer, Vehiculo vehiculo) throws PedidoInexistenteException, ChoferNoDisponibleException, VehiculoNoDisponibleException, VehiculoNoValidoException, ClienteConViajePendienteException {  
 if (!this.pedidos.containsValue(pedido)) { **#1**  
 throw new PedidoInexistenteException(pedido); **#2**  
 } else if (!this.choferesDesocupados.contains(chofer)) { **#3**  
 throw new ChoferNoDisponibleException(chofer); **#4**  
 } else if (!this.vehiculosDesocupados.contains(vehiculo)) { **#5**  
 throw new VehiculoNoDisponibleException(vehiculo); **#6**  
 } else if (vehiculo.getPuntajePedido(pedido) == null) { **#7**  
 throw new VehiculoNoValidoException(vehiculo, pedido); **#8**  
 } else {  
 Viaje viaje = new Viaje(pedido, chofer, vehiculo); **#9**  
 this.pedidos.remove(pedido.getCliente()); **#10**  
 this.viajesIniciados.put(viaje.getPedido().getCliente(), viaje); **#11**  
 this.choferesDesocupados.remove(chofer); **#12**  
 this.vehiculosDesocupados.remove(vehiculo); **#13**  
 }  
}

**Grafo de Flujo**

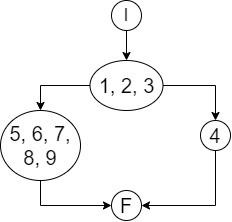


**Clase: Empresa**

**Método: pagarYFinalizarViaje**

public void pagarYFinalizarViaje(int calificacion) throws ClienteSinViajePendienteException {  
 Cliente cli = (Cliente)this.usuarioLogeado; **#1**  
 Viaje viaje = this.getViajeDeCliente(cli); **#2**  
 if (viaje == null) { **#3**  
 throw new ClienteSinViajePendienteException(); **#4**  
 } else {  
 this.viajesTerminados.add(viaje); **#5**  
 **#6** this.viajesIniciados.remove(viaje.getPedido().getCliente());   
 viaje.finalizarViaje(calificacion); **#7**  
 this.choferesDesocupados.add(viaje.getChofer()); **#8**  
 this.vehiculosDesocupados.add(viaje.getVehiculo()); **#9**  
 }  
}

**Grafo de Flujo**

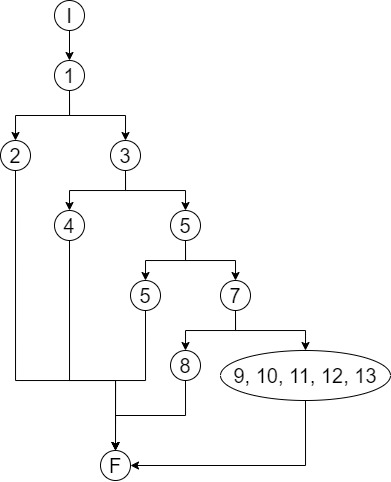


**Clase: Empresa**

**Método: crearViaje**

public void crearViaje(Pedido pedido, Chofer chofer, Vehiculo vehiculo) throws PedidoInexistenteException, ChoferNoDisponibleException, VehiculoNoDisponibleException, VehiculoNoValidoException, ClienteConViajePendienteException {  
 if (!this.pedidos.containsValue(pedido)) { **#1**  
 throw new PedidoInexistenteException(pedido); **#2**  
 } else if (!this.choferesDesocupados.contains(chofer)) { **#3**  
 throw new ChoferNoDisponibleException(chofer); **#4**  
 } else if (!this.vehiculosDesocupados.contains(vehiculo)) { **#5**  
 throw new VehiculoNoDisponibleException(vehiculo); **#6**  
 } else if (vehiculo.getPuntajePedido(pedido) == null) { **#7**  
 throw new VehiculoNoValidoException(vehiculo, pedido); **#8**  
 } else {  
 Viaje viaje = new Viaje(pedido, chofer, vehiculo); **#9**  
 this.pedidos.remove(pedido.getCliente()); **#10**  
 this.viajesIniciados.put(viaje.getPedido().getCliente(), viaje); **#11**  
 this.choferesDesocupados.remove(chofer); **#12**  
 this.vehiculosDesocupados.remove(vehiculo); **#13**  
 }  
}

**Grafo de Flujo**

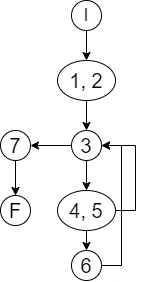


**Clase: Empresa**

**Método: validarPedido**

public boolean validarPedido(Pedido pedido) {  
 Iterator<Vehiculo> it = this.iteratorVehiculos(); **#1**  
 Vehiculo vehiculoValido = null; **#2**  
  
 while(it.hasNext() && vehiculoValido == null) { **#3**  
 Vehiculo actual = (Vehiculo)it.next();**#4**  
 if (actual.getPuntajePedido(pedido) != null) { **#5**  
 vehiculoValido = actual; **#6**  
 }  
 }  
  
 return vehiculoValido != null; **#7**  
}

**Grafo de Flujo**



**Clase: Empresa**

**Método: logout**

public void logout() {  
 this.usuarioLogeado = null; **#1**  
}

**Grafo de Flujo**

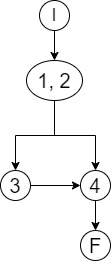


**Clase: Moto**

**Método: getPuntajePedido**

public Integer getPuntajePedido(Pedido pedido) {  
 Integer respuesta = null; **#1**  
 if (!pedido.isMascota() && !pedido.isBaul() && pedido.getCantidadPasajeros() == 1) { **#2**  
 respuesta = 1000; **#3**  
 }  
  
 return respuesta; **#4**  
}

**Grafo de Flujo**

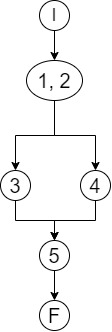


**Clase: Auto**

**Método: getPuntajePedido**

public Integer getPuntajePedido(Pedido pedido) {  
 Integer respuesta; **#1**  
 if ((!pedido.isMascota() || this.isMascota()) && pedido.getCantidadPasajeros() <= this.getCantidadPlazas()) { **#2**  
 respuesta = pedido.getCantidadPasajeros() \* 40; **#3**  
 } else {  
 respuesta = null; **#4**  
 }  
  
 return respuesta; **#5**  
}

**Grafo de Flujo**

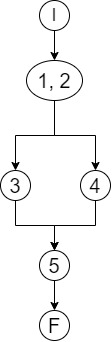


**Clase: Combi**

**Método: getPuntajePedido**

public Integer getPuntajePedido(Pedido pedido) {  
 Integer respuesta; **#1**  
 if (pedido.isMascota() && !this.isMascota()) { **#2**  
 respuesta = null; **#3**  
 } else {  
 respuesta = pedido.getCantidadPasajeros() \* 10; **#4**  
 }  
  
 return respuesta; **#5**  
}

**Grafo de Flujo**

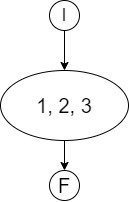


**Clase: ChoferPermanente**

**Método: getSueldoBruto**

public double getSueldoBruto() {  
 double ant = 0.05 \* (double)this.getAntiguedad(); **#1**  
 double h = 0.07 \* (double)this.cantidadHijos; **#2**  
 return Chofer.getSueldoBasico() \* (1.0 + ant + h); **#3**  
}

**Grafo de Flujo**

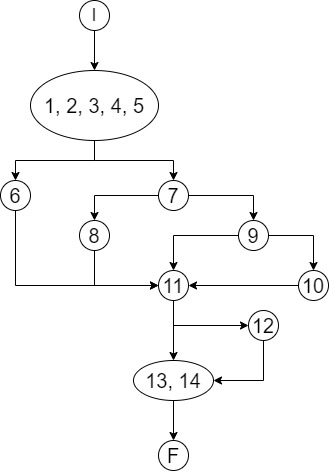


**Clase: Viaje**

**Método: getValor**

public double getValor() {  
 double resultado = 0.0; **#1**  
 double incrementoZona = 0.0; **#2**  
 double incrementoBaul = 0.0; **#3**  
 double incrementoMascota = 0.0; **#4**  
 if (this.pedido.getZona().equals("ZONA\_PELIGROSA")) { **#5**  
 incrementoZona = 0.1 \* (double)this.pedido.getCantidadPasajeros() + 0.2 \* (double)this.pedido.getKm(); **#6**  
 } else if (this.pedido.getZona().equals("ZONA\_SIN\_ASFALTAR")){**#7**   
 incrementoZona = 0.2 \* (double)this.pedido.getCantidadPasajeros() + 0.15 \* (double)this.pedido.getKm(); **#8**  
 } else if (this.pedido.getZona().equals("ZONA\_SIN\_ASFALTAR")){**#9**  
 incrementoZona = 0.2 \* (double)this.pedido.getCantidadPasajeros() + 0.15 \* (double)this.pedido.getKm(); **#10**  
 }  
  
 if (this.pedido.isMascota()) { **#11**  
 incrementoMascota = 0.1 \* (double)this.pedido.getCantidadPasajeros() + 0.2 \* (double)this.pedido.getKm(); **#12**  
 }  
  
 resultado = (1.0 + incrementoZona + incrementoMascota + incrementoBaul) \* *valorBase*; **#13**  
 return resultado; **#14**  
}

**Grafo de Flujo**

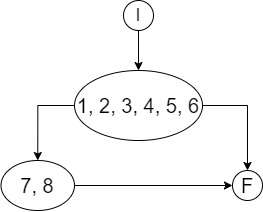


**Clase: Controlador**

**Método: nuevoViaje**

public void nuevoViaje() {  
 Pedido pedido = this.vista.getPedidoSeleccionado(); **#1**  
 Chofer chofer = this.vista.getChoferDisponibleSeleccionado(); **#2**  
 Vehiculo vehiculo = this.vista.getVehiculoDisponibleSeleccionado(); **#3**  
  
 try { **#4**  
 **#5** Empresa.getInstance().crearViaje(pedido, chofer, vehiculo);   
 this.vista.actualizar(); **#6**  
 } catch (ChoferNoDisponibleException | VehiculoNoDisponibleException | VehiculoNoValidoException | ClienteConViajePendienteException | PedidoInexistenteException var5) {  
 Exception e = var5; **#7**  
 **#8** this.vista.getOptionPane().ShowMessage(((Exception)e).getMessage());  
 }  
  
}

**Grafo de Flujo**

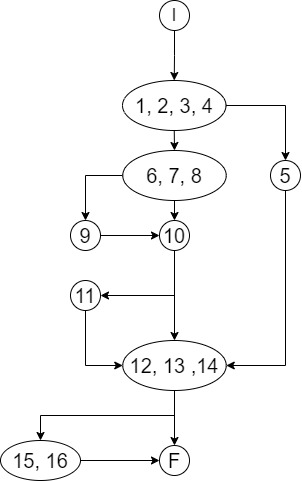


**Clase: Controlador**

**Método: nuevoVehiculo**

public void nuevoVehiculo() {  
 String tipo = this.vista.getTipoVehiculo(); **#1**   
 String patente = this.vista.getPatente(); **#2**  
 Vehiculo v = null; **#3**  
 if (tipo.equals("MOTO")) { **#4**  
 v = new Moto(patente); **#5**  
 } else {   
 int plazas = this.vista.getPlazas(); **#6**  
 boolean mascota = this.vista.isVehiculoAptoMascota(); **#7**  
 if (tipo.equals("AUTO")) { **#8**  
 v = new Auto(patente, plazas, mascota); **#9**  
 }  
  
 if (tipo.equals("COMBI")) { **#10**  
 v = new Combi(patente, plazas, mascota); **#11**  
 }  
 }  
  
 try { **#12**  
 Empresa.getInstance().agregarVehiculo((Vehiculo)v); **#13**  
 this.vista.actualizar(); **#14**  
 } catch (VehiculoRepetidoException var6) {  
 VehiculoRepetidoException e = var6; **#15**  
 this.vista.getOptionPane().ShowMessage(e.getMessage()); **#16**  
 }  
  
}

**Grafo de Flujo**

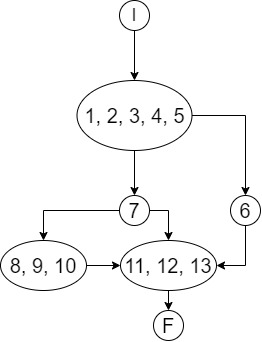


**Clase: Controlador**

**Método: nuevoChofer**

public void nuevoChofer() {  
 String tipo = this.vista.getTipoChofer(); **#1**  
 String nombre = this.vista.getNombreChofer(); **#2**  
 String dni = this.vista.getDNIChofer(); **#3**  
 Chofer chofer = null; **#4**  
 if (tipo.equalsIgnoreCase("TEMPORARIO")) { **#5**  
 chofer = new ChoferTemporario(dni, nombre); **#6**  
 } else if (tipo.equalsIgnoreCase("PERMANENTE")) { **#7**  
 int anio = this.vista.getAnioChofer(); **#8**  
 int hijos = this.vista.getHijosChofer(); **#9**  
 chofer = new ChoferPermanente(dni, nombre, anio, hijos); **#10**  
 }  
  
 try { **#11**  
 Empresa.getInstance().agregarChofer((Chofer)chofer); **#12**  
 this.vista.actualizar(); **#13**  
 } catch (ChoferRepetidoException var7) {  
 }  
  
}

**Grafo de Flujo**

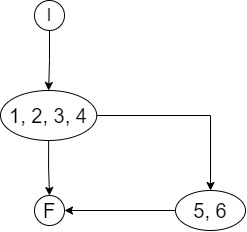


**Clase: Controlador**

**Método: calificarPagar**

public void calificarPagar() {  
 int calificacion = this.vista.getCalificacion(); **#1**  
  
 try { **#2**   
 Empresa.getInstance().pagarYFinalizarViaje(calificacion); **#3**  
 this.vista.actualizar(); **#4**  
 } catch (ClienteSinViajePendienteException var3) {  
 ClienteSinViajePendienteException e = var3; **#5**  
 this.vista.getOptionPane().ShowMessage(e.getMessage()); **#6**  
 }  
  
}

**Grafo de Flujo**

****

**Clase: Controlador**

**Método: login**

public void login() {  
 try { **#1**  
 String usserName = this.vista.getUsserName(); **#2**  
 String pass = this.vista.getPassword(); **#3**  
 Empresa.getInstance().login(usserName, pass); **#4**  
 this.vista.logearUsuario(); **#5**  
 } catch (PasswordErroneaException | UsuarioNoExisteException var3) {  
 Exception e = var3; **#6**  
 **#7**this.vista.getOptionPane().ShowMessage(((Exception)e).getMessage());  
 }  
  
}

**Grafo de Flujo**

**A diagram of a number system

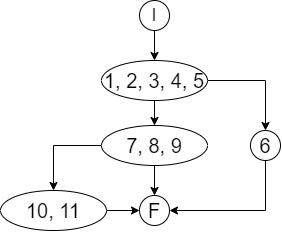
Description automatically generated**

**Clase: Controlador**

**Método: registrar**

public void registrar() {  
 String nombreReal = this.vista.getRegNombreReal(); **#1**  
 String nombreUsuario = this.vista.getRegUsserName(); **#2**  
 String pass = this.vista.getRegPassword(); **#3**  
 String confirm = this.vista.getRegConfirmPassword(); **#4**  
 if (!pass.equals(confirm)) { **#5**  
 this.vista.getOptionPane().ShowMessage(Mensajes.PASS\_NO\_COINCIDE.getValor()); **#6**  
 } else {  
 try { **#7**  
 Empresa.getInstance().agregarCliente(nombreUsuario, pass, nombreReal); **#8**  
 this.escribir(); **#9**  
 } catch (UsuarioYaExisteException var6) {  
 UsuarioYaExisteException e = var6; **#10**  
 **#11** this.vista.getOptionPane().ShowMessage(e.getMessage());  
 }  
 }  
  
}

**Grafo de Flujo**

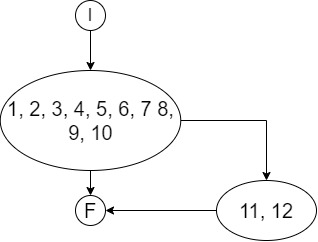


**Clase: Controlador**

**Método: nuevoPedido**

public void nuevoPedido() {  
 Cliente cliente = (Cliente)Empresa.getInstance().getUsuarioLogeado(); **#1**  
 int cantidadPasajeros = this.vista.getCantidadPax(); **#2**  
 boolean mascota = this.vista.isPedidoConMascota(); **#3**  
 boolean baul = this.vista.isPedidoConBaul(); **#4**  
 int km = this.vista.getCantKm(); **#5**  
 String zona = this.vista.getTipoZona(); **#6**  
 Pedido pedido = new Pedido(cliente, cantidadPasajeros, mascota, baul, cantidadPasajeros, zona); **#7**  
  
 try { **#8**  
 Empresa.getInstance().agregarPedido(pedido); **#9**  
 this.vista.actualizar(); **#10**  
 } catch (ClienteConViajePendienteException | ClienteConPedidoPendienteException | ClienteNoExisteException | SinVehiculoParaPedidoException var9) {  
 Exception e = var9; **#11**  
 **#12**this.vista.getOptionPane().ShowMessage(((Exception)e).getMessage());  
 }  
  
}

**Grafo de Flujo**



**Test de Integración Orientado a Objetos**

**Caso de Uso 1**

**Descripción**

Un cliente debe iniciar sesión para poder utilizar el sistema, haciendo login con su usuario y password. De no tener tales credenciales, debe registrarse en el sistema para obtener una cuenta.

Un cliente debe iniciar sesión para poder utilizar el sistema, haciendo login con su usuario y password. De no tener tales credenciales, debe registrarse en el sistema para obtener una cuenta.

**Actores**

Cliente a través de la interfaz de usuario gráfica.

**Pre-condiciones**

El cliente debe estar registrado en el sistema.

**Flujo Normal**

1. El cliente accede al sistema.
2. El sistema solicita credenciales.
3. El cliente ingresa escribiendo nombre de usuario y password.
4. El sistema valida las credenciales del cliente y le da la bienvenida.

**Flujo Alternativo 1**

1. El cliente no tiene una cuenta por lo que decide registrarse en el sistema.
2. El sistema solicita los datos necesarios para crear la cuenta al cliente.
3. El cliente ingresa los datos.
4. El sistema crea la cuenta del cliente.

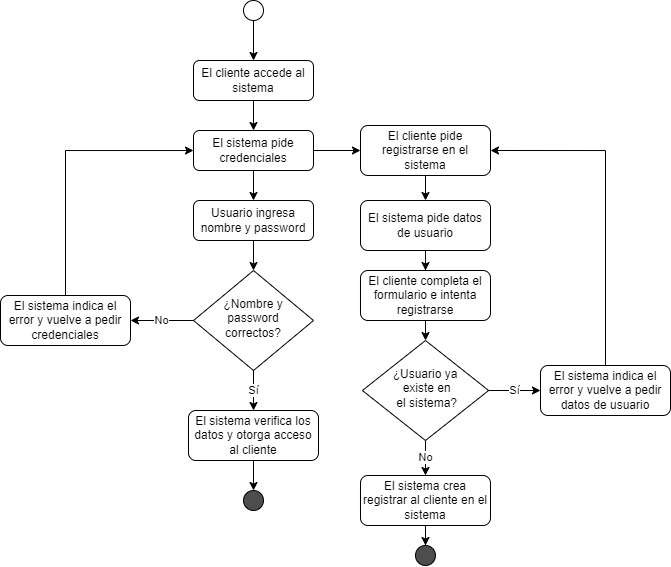
**Excepciones**

* E1. Password errónea al intentar hacer login. El sistema notifica el error.
* E2. El usuario ingresado no esta registrado en el sistema. El sistema notifica el error.
* E3. El nombre de usuario a registrar ya existe en el sistema. El sistema notifica el error.

**Post-condiciones**

1. El usuario obtiene acceso al sistema.

**Diagrama de actividad**



**Observaciones**

* Al momento de registrarse, si las contraseñas no coinciden se indicará el error al usuario pero esto no lanzará una excepción, por lo que no se lo tomará en cuenta en el diagrama de actividad ni en los casos de prueba.

**Casos de Prueba**

**Caso de Uso 2**

**Descripción**

Un cliente necesita transportarse, por lo que utiliza el sistema para solicitar un viaje.

**Actores**

Cliente a través de la interfaz de usuario gráfica.

**Pre-condiciones**

El cliente debe haber iniciado sesión.

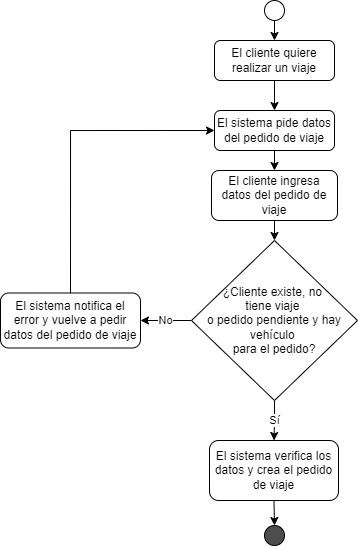
**Flujo Normal**

1. El cliente realiza un pedido.
2. El sistema solicita datos del pedido.
3. El cliente ingresa los datos del pedido.
4. El sistema crea un pedido, asignándole un chofer, vehículo y cliente.

**Excepciones**

* E1. El cliente que realizó el pedido no existe en el sistema. El sistema notifica el error.
* E2. El cliente ya tiene un viaje pendiente que debe resolver antes de poder realizar otro pedido. El sistema notifica el error.
* E3. El cliente tiene un pedido pendiente que debe resolver antes de poder realizar otro pedido. El sistema notifica el error.
* E4. No existe vehículo que cumpla con los requisitos del pedido. El sistema notifica el error.

**Diagrama de actividad**



**Casos de Prueba**

**Caso de Uso 3**

**Descripción**

El cliente termina llega a destino y debe finalizar el viaje en el sistema realizando el pago.

**Actores**

El cliente a través de la interfaz de usuario gráfica.

**Pre-condiciones**

El cliente debe haber iniciado sesión y ya tiene un viaje pendiente.

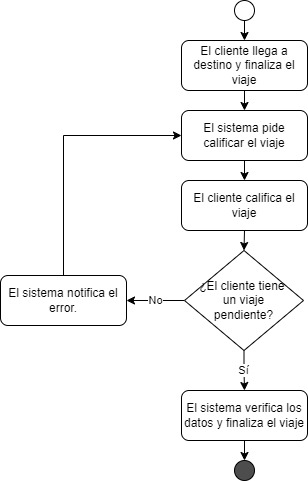
**Flujo Normal**

1. El cliente finaliza el viaje.
2. El sistema solicita una calificación del viaje.
3. El cliente ingresa la calificación y realiza el pago.
4. El sistema agrega el viaje a la lista de viajes terminados, desocupa al chofer y al vehículo, y quita de la lista de viajes iniciados al cliente.

**Excepciones**

* E1. El cliente no tiene un viaje pendiente que finalizar. El sistema notifica el error.

**Diagrama de actividad**

****

**Casos de Prueba**

**Caso de Uso 4**

**Descripción**

El administrador contrata a un nuevo chofer y le da el alta en el sistema.

**Actores**

El administrador a través del panel de administrador.

**Pre-condiciones**

El administrador debe haber iniciado sesión en el sistema.

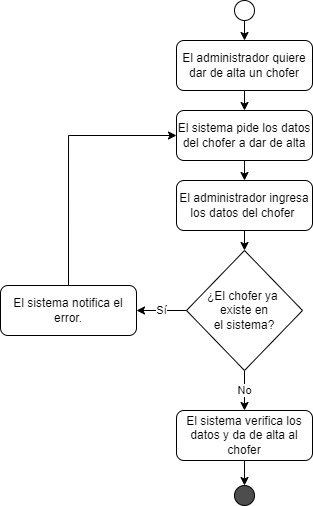
**Flujo Normal**

1. El administrador da de alta un chofer.
2. El sistema pide datos del chofer.
3. El administrador ingresa datos del chofer.
4. El sistema agrega el chofer a la lista de choferes y a la de choferes desocupados.

**Excepciones**

* E1. El chofer ya está existe en el sistema.

**Diagrama de actividad**

****

**Casos de Prueba**

**Caso de Uso 5**

**Descripción**

El administrador quiere de dar de alta un vehículo en el sistema.

**Actores**

El administrador a través del panel de administrador.

**Pre-condiciones**

El administrador debe haber iniciado sesión en el sistema.

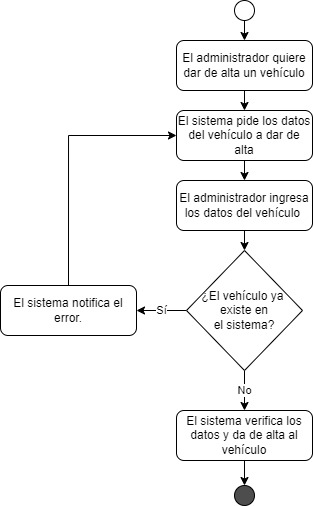
**Flujo Normal**

1. El administrador quiere dar de alta un vehículo
2. El sistema pide datos del vehículo
3. El administrador ingresa los datos del vehículo
4. El sistema agrega el vehículo a la lista de vehículos y a la de vehículos desocupados.

**Excepciones**

* E1. El vehículo ya existe en el sistema. El sistema notifica el error al administrador.

**Diagrama de actividades**

****

**Casos de Prueba**

**Caso de Uso 6**

**Descripción**

El administrador asigna un vehículo y un chofer al pedido pendiente del cliente para comenzar el viaje.

**Actores**

El administrador a través del panel de administrador.

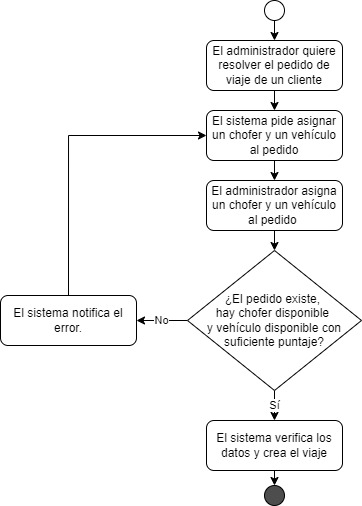
**Flujo Normal**

1. Asas

**Excepciones**

* E1. No hay choferes disponibles para realizar el viaje. El sistema notifica el error.
* E2. No hay vehículos disponibles para realizar el viaje. El sistema notifica el error.
* E3. El vehículo no es válido para el viaje debido a que su puntuación es menor a la esperada por el cliente. El sistema notifica el error.
* E4. El pedido no existe en el sistema. El sistema notifica el error.
* E5. El cliente ya tiene un viaje pendiente.

**Diagrama de actividad**

****

**Casos de prueba**