

Taller de Programación I

Informe Trabajo Práctico Final

Grupo 7

Palladino, Sofía Isabella

Delgado, Facundo

Parise, Thiago

Gonzalez, Facundo Nehuen

Introducción:

Este informe presenta los resultados de las pruebas realizadas sobre el sistema de gestión de pedidos de vehículos para viajes, evaluando tanto la capa de datos como la de negocio, así como la interfaz gráfica de usuario (GUI). Las pruebas se llevaron a cabo principalmente mediante métodos de caja negra, utilizando el marco de pruebas JUnit 4 para asegurar la correcta funcionalidad del sistema de extremo a extremo.

El objetivo principal de estas pruebas fue validar que los métodos en las clases de modelo de datos y de negocio respondan correctamente a distintos casos de uso y gestionen adecuadamente las posibles excepciones. Adicionalmente, los test de GUI se ejecutaron para comprobar la correcta interacción y respuesta de la interfaz frente a diferentes entradas del usuario, verificando la activación de botones y mensajes de error cuando se cumplen ciertas condiciones de validación.

A lo largo del informe, se detallan los errores detectados en cada componente del sistema, dividiéndolos en errores específicos del modelo de datos, del modelo de negocio, del controlador, y finalmente los problemas de la interfaz gráfica.

Recursos:

- **Repositorio de GitHub:** https://github.com/nehuengonz/2024_grupo_7
- **Tablas de particiones, Baterías de prueba y Escenarios en Hoja de cálculo:**  Tabla de Particiones - TallerDeProgramacion1-GRUPO-7

Errores detectados

A continuación se listan los errores detectados por paquete de la aplicación. **Para ver todos los tests ir al siguiente link de google:**

[Tabla de Particiones - TallerDeProgramacion1-GRUPO-7](#)

Modelo de Datos:

- **ChoferPermanente.getSueldoBruto():** En la clase ChoferPermanente el sueldo bruto obtenido del método getSueldoBruto() no se calcula correctamente al tener 20 años de antigüedad. Por consecuencia el método getSueldoNeto() también retorna mal el resultado.
En la clase ChoferPermanente no lanza excepción cuando el año de ingreso es mayor al año actual.

| Tabla de Particiones | | | |
|----------------------|---|--------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| Chofer | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} < 20 \ \&\& \ \text{cantHijos} = 0$ | si | 1 |
| | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} < 20 \ \&\& \ \text{cantHijos} > 0$ | si | 2 |
| | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} = 20 \ \&\& \ \text{cantHijos} = 0$ | si | 3 |
| | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} = 20 \ \&\& \ \text{cantHijos} > 0$ | si | 4 |
| | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} = 0 \ \&\& \ \text{cantHijos} = 0$ | si | 5 |
| | $1900 \leq \text{anioIngreso} \leq 3000 \ \&\& \ \text{Año Actual} - \text{anioIngreso} = 0 \ \&\&$ | si | 6 |

| | | | |
|--|-------------------------|--|--|
| | <i>cantHijos > 0</i> | | |
|--|-------------------------|--|--|

| Batería de Pruebas | | | | |
|--------------------|----------------------|--------------------|---|-----------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada | clases de equiv. que abarca |
| 1 | <i>dni</i> | <i>{"1234567"}</i> | <i>El sueldo bruto se calcula correctamente</i> | 1 |
| | <i>nombre</i> | <i>{"Facundo"}</i> | | |
| | <i>anioIngreso</i> | 2018 | | |
| | <i>cantidadHijos</i> | 0 | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| | | | | |
| 2 | <i>dni</i> | <i>{"1234567"}</i> | <i>El sueldo bruto se calcula correctamente</i> | 2 |
| | <i>nombre</i> | <i>{"Facundo"}</i> | | |
| | <i>anioIngreso</i> | 2018 | | |
| | <i>cantidadHijos</i> | 1 | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| | | | | |
| 3 | <i>dni</i> | <i>{"1234567"}</i> | 4000.0 | 3 |
| | <i>nombre</i> | <i>{"Facundo"}</i> | | |
| | <i>anioIngreso</i> | 2000 | | |
| | <i>cantidadHijos</i> | 0 | | |
| | | | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| 4 | <i>dni</i> | <i>{"1234567"}</i> | 4420.0 | 4 |
| | <i>nombre</i> | <i>{"Facundo"}</i> | | |
| | <i>anioIngreso</i> | 2000 | | |
| | <i>cantidadHijos</i> | 3 | | |
| | | | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| 5 | <i>dni</i> | <i>{"1234567"}</i> | <i>El sueldo bruto se calcula correctamente</i> | 5 |
| | <i>nombre</i> | <i>{"Facundo"}</i> | | |
| | <i>anioIngreso</i> | 2024 | | |

| | | | | |
|---|---------------------|-------------|--|---|
| | cantidadHijos | 0 | | |
| | | | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| | | | | |
| 6 | dni | {"1234567"} | El sueldo bruto se calcula correctamente | 6 |
| | nombre | {"Facundo"} | | |
| | anioIngreso | 2024 | | |
| | cantidadHijos | 3 | | |
| | | | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |
| 7 | dni | {"1234567"} | El calculo de la antigüedad debería arrojar alguna excepción si es mayor al año actual | 7 |
| | nombre | {"Facundo"} | | |
| | anioIngreso | 3000 | | |
| | cantidadHijos | 0 | | |
| | | | | |
| | | | | |
| | Chofer.sueldoBasico | 2000.0 | | |

- **Viaje.getValor():** No se calcula bien el valor del viaje sin extras y estandar. No se calcula bien el valor del viaje con mascota estandar. No se calcula bien el valor del viaje con baul sin asfaltar.

| Tabla de Particiones | | | |
|----------------------|--|--------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| N/A | | | |

| Batería de Pruebas | | | |
|--------------------|------------------|--|-----------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | Pedido | {{"Roberto","123","Roberto"}, 1, false,false,10,"ZONA_STANDARD"} | 10.5 |

| | | | |
|---|--------|--|------|
| 2 | pedido | {cliente, 4, false, false, 10, Constantes.ZONA_PELIGROSA} | 17.0 |
| 3 | pedido | {cliente, 1, false, false, 10, Constantes.ZONA_SIN_ASFALTAR} | 13.5 |
| 4 | pedido | {cliente, 1, true, false, 10, Constantes.ZONA_STANDARD} | 21.0 |
| 5 | pedido | {cliente, 4, true, false, 10, Constantes.ZONA_PELIGROSA} | 29.0 |
| 6 | pedido | {cliente, 4, false, true, 10, Constantes.ZONA_SIN_ASFALTAR} | 21.0 |

- **Combi.getPuntajePedido (Pedido pedido):** En la clase Combi el puntaje obtenido del método getPuntajePedido(Pedido pedido) no se calcula correctamente cuando pedido.isBaul()==true.

| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
|-----------------|--|--------------------------------|--|
| pedido | pedido != null | Si | 1 |
| | pedido = null | No | 2 |
| | pedido.isMascota() = true, vehiculo.isMascota() = true (vehículo acepta mascotas) | Si | 3 |
| | pedido.isMascota() = true, vehiculo.isMascota() = false (vehículo no acepta mascotas) | Si | 4 |
| | pedido.isMascota() = false, vehiculo.isMascota() = false (vehículo no acepta mascotas) | Si | 5 |
| | pedido.isMascota() = false, vehiculo.isMascota() = true (vehículo acepta mascotas) | Si | 6 |
| | pedido.isBaul() = true | Si | 7 |
| | pedido.isBaul() = false | Si | 8 |
| | 4<pedido.CantidadPasajeros<=combi.CantidadPlazas | Si | 9 |
| | cantidadPasajeros > capacidad del vehículo | Si | 10 |
| | pedido.CantidadPasajeros<=5 | Si | 11 |

| Batería de Pruebas | | | | |
|--------------------|------------------|--|----------------------|-----------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada | clases de equív. que abarca |
| 1 | pedido | (cliente, 6, true, false, 25, Constantes.ZONA_SIN_ASFALTAR) con Combi.PetFriendly=true | 60 | 1, 3, 8, 9 |
| 2 | pedido | (cliente, 6, false, false, 25, Constantes.ZONA_SIN_ASFALTAR) con Combi.PetFriendly=false | 60 | 1, 5, 8, 9 |
| 3 | pedido | (cliente, 6, true, false, 25, Constantes.ZONA_SIN_ASFALTAR) con Combi.PetFriendly=false | null | 1, 4, 8, 9 |
| 4 | pedido | (cliente, 6, true, true, 25, Constantes.ZONA_SIN_ASFALTAR) con Combi.PetFriendly=true | 160 (salida real 60) | 1, 3, 7, 9 |
| 5 | pedido | (cliente, 6, false, true, 25, Constantes.ZONA_SIN_ASFALTAR) | 160 (salida real 60) | 1, 6, 7, 9 |
| 6 | pedido | (cliente, 13, false, true, 25, | null | 1, 6, 7, 10 |

| | | | | |
|---|--------|---|------|----------------|
| | | Constantes.ZONA_SIN _ASFALTAR) | | |
| 7 | pedido | (cliente,3 , false, true, 25, Constantes.ZONA_SIN _ASFALTAR) | null | 1, 6, 7, 11 |


- **Auto.getPuntajePedido (Pedido pedido):** En la clase Auto el puntaje se calcula erróneamente en el método getPuntajePedido(Pedido pedido) cuando el pedido no solicita uso de baúl y el vehículo satisface el pedido.

| Tabla de Particiones | | | | |
|----------------------|--|--------------------------------|--|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia | |
| Pedido | <i>pedido != null, pedido solicita uso de baul y vehiculo satisface pedido</i> | <i>si</i> | 1 | |
| | <i>pedido != null, pedido no solicita uso de baul y vehiculo satisface pedido</i> | <i>si</i> | 2 | |
| | <i>pedido != null, pedido solicita uso de baul y vehiculo no satisface las necesidades del pedido</i> | <i>si</i> | 3 | |
| | <i>pedido != null, pedido no solicita uso de baul y vehiculo no satisface las necesidades del pedido</i> | <i>si</i> | 4 | |
| | <i>pedido == null</i> | <i>no</i> | 5 | |

| Batería de Pruebas | | | | |
|--------------------|------------------|--|-----------------|-----------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada | clases de equiv. que abarca |
| 1 | pedido | { { "wutang", "12345", "Thiago" }, 4, true, true, 10, Constantes.ZONA_PELIGROSA } | 160 | 1 |
| 2 | pedido | { { "wutang", "12345", "Thiago" }, 4, true, false, 10, Constantes.ZONA_PELIGROSA } | 120 | 2 |
| 3 | pedido | { { "wutang", "12345", "Thiago" }, 5, true, true, 10, Constantes.ZONA_PELIGROSA } | null | 3 |
| 4 | pedido | { { "wutang", "12345", "Thiago" }, 5, true, false, 10, Constantes.ZONA_PELIGROSA } | null | 4 |

Modelo de Negocio:

A continuación se listan algunos de los escenarios de la clase Escenario utilizados en los testeos. **Para ver en detalle que tiene cada escenario ir a la hoja Escenarios Empresa de la hoja de cálculo:**

 Tabla de Particiones - TallerDeProgramacion1-GRUPO-7

| |
|---|
| Base (Todas las colecciones vacías) |
| 1 (Escenario con clientes, no se usa al final) |
| 2 (Escenario con las colecciones llenas, menos viajesIniciados y viajesTerminados) |
| 3 (Escenario con las colecciones llenas, con un viaje viajesIniciados y sin viajesTerminados) |
| 4 (Escenario con las colecciones llenas, sin pedidos, con un viaje iniciado y uno finalizado) |
| 5 (Escenario que se usa para validar pedidos) |
| 6 (Escenario con las colecciones Clientes, Choferes, Vehiculos, ChoferesDesocupados y Vehiculos Desocupados llenas) |

- **crearViaje(Pedido pedido, Chofer chofer, Vehiculo vehiculo):** Se encontró que la excepción ChoferNoDisponibleException no se construye bien cuando se crea un viaje con un chofer no disponible.
El método sigue ejecutando cuando debería arrojar ClienteConViajePendienteException

| Tabla de Particiones | | | |
|----------------------|--|--------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| Pedido | pedido != null | Si | 1 |
| | pedido = null | No | 2 |
| Chofer | chofer != null | Si | 3 |
| | chofer= null | No | 4 |
| Vehiculo | vehiculo != null | Si | 5 |
| | vehiculo = null | No | 6 |

| Batería de Pruebas - Escenario 2 | | | |
|----------------------------------|------------------|--|-----------------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | pedido | ("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | Viaje creado con éxito |
| | chofer | ("1234567","Roberto",2020,0) | |
| | vehiculo | ("abc123",4,true) | |
| 2 | pedido | ("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | VehiculoNoValidoException |
| | chofer | ("1234567","Roberto",2020,0) | |
| | vehiculo | ("pat333") | |
| 3 | pedido | ("facundo","123","Facundo"), 4, true, true, 10, Constantes.ZONA_STANDARD); | PedidoInexistentException |
| | chofer | ("1234567","Roberto",2020,0) | |
| | vehiculo | ("abc123",4,true) | |
| Batería de Pruebas - Escenario 3 | | | |
| 4 | pedido | ("thiago","321","Thiago"), 4, true, true, 10, Constantes.ZONA_STANDARD); | ClienteConViajePendienteException |
| | chofer | ("1234568","Alberto",2019,3) | |
| | vehiculo | ("abc123",4,true) | |
| 5 | pedido | ("thiago","321","Thiago"),1,false,false,3,STANDARD); ("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | ChoferNoDisponibleException |
| | chofer | ("1234567","Roberto",2020,0) | |
| | vehiculo | "pat333"; "abc123" | |
| 6 | pedido | ("thiago","321","Thiago"),1,false,false,3,STANDARD); ("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | VehiculoNoDisponibleException |
| | chofer | ("1234567","Roberto",2020,0); ("1234568","Alberto",2019,3) | |

- **agregarPedido(Pedido pedido):** El sistema no arroja la excepción `ClienteConViajePendienteException` cuando el cliente tiene un viaje en curso.

| Tabla de Particiones | | | |
|----------------------|--|-----------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| Pedido | El pedido es válido y la empresa puede satisfacerlo | Si | 1 |
| | El pedido es válido y la empresa no cumple con las condiciones para satisfacerlo | Si | 2 |
| | El cliente tiene un viaje iniciado | Si | 3 |
| | El cliente tiene un pedido iniciado | Si | 4 |
| | El atributo cliente no es un cliente registrado en la empresa | Si | 5 |
| | El pedido es nulo | No | 6 |

| Batería de Pruebas - Escenario 4 | | | |
|----------------------------------|------------------|---------------------------------|-----------------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | Pedido | { "thiago", "321", "Thiago" } | Pedido creado exitosamente |
| | | 1 | |
| | | TRUE | |
| | | TRUE | |
| | | 2 | |
| | | Constantes.ZONA_PELIGROSA | |
| 2 | Pedido | { "thiago", "321", "Thiago" } | SinVehiculoPara PedidoException |
| | | 8 | |
| | | TRUE | |
| | | TRUE | |
| | | 10 | |
| | | Constantes.ZONA_PELIGROSA | |
| 3 | Pedido | { "facundo", "123", "Facundo" } | ClienteConViajePendienteException |
| | | 1 | |
| | | TRUE | |
| | | TRUE | |
| | | 2 | |
| | | Constantes.ZONA_PELIGROSA | |

| | | | |
|---|--------|--------------------------------------|------------------------------------|
| 4 | Pedido | { "thiago", "321", "Thiago" } | ClienteConPedidoPendienteException |
| | | 1 | |
| | | TRUE | |
| | | TRUE | |
| | | 2 | |
| 5 | Pedido | Constantes.ZONA_PELIGROSA | ClienteNoExisteException |
| | | { "usuarioNuevo", "123", "Usuario" } | |
| | | 1 | |
| | | TRUE | |
| | | TRUE | |
| | | 2 | |
| | | Constantes.ZONA_PELIGROSA | |

- **login(String usserName, String pass):** Al querer loguear un Administrador y se pone mal la contraseña no arroja la excepcion PasswordErroneaException en cambio arroja UsuarioNoExisteException

| Tabla de Particiones | | | |
|----------------------|--|--------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| usserName | username != null | Si | 1 |
| | username != "" | Si | 2 |
| | username = null | No | 3 |
| | username = "" | No | 4 |
| pass | pass != null | Si | 6 |
| | pass != "" | Si | 7 |
| | pass = null | No | 8 |
| | pass = "" | No | 9 |

| Batería de Pruebas | | | |
|--------------------|------------------|-------|----------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | username | admin | Usuario logueado con éxito |
| | pass | admin | |
| | | | |
| | | | |
| 2 | username | a | UsuarioNoExisteException |
| | pass | admin | |

| | | | |
|-------------------------------------|------------------|------------|----------------------------------|
| 3 | ussername | admin | PasswordErrone aException |
| | pass | 123 | |
| 4 | ussername | a | UsuarioNoExiste Exception |
| | pass | 123 | |
| 5 | ussername | ADMIN | UsuarioNoExiste Exception |
| | pass | admin | |
| 6 | ussername | admin | PasswordErrone aException |
| | pass | ADMIN | |
| 7 | ussername | Admin | UsuarioNoExiste Exception |
| | pass | admin | |
| 8 | ussername | admin | PasswordErrone aException |
| | pass | Admin | |
| Batería de Pruebas - Escenario 2 | | | |
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | ussername | "facundo", | Usuario logueado con exito |
| | pass | "123" | |
| | | | |
| | | | |
| 2 | ussername | "thiago" | PasswordErrone aException |
| | pass | "1234" | |
| 3 | ussername | "nehu" | UsuarioNoExiste Exception |
| | pass | "4321" | |

- **validarPedido(Pedido pedido):** Según el escenario planteado (Escenario 5) un pedido de 10 pasajeros no podría satisfacerse con ningún vehículo cargado en la Empresa.

| Tabla de Particiones | | | |
|-----------------------------|---|---|---|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| Pedido | pedido != null | <i>Si</i> | <i>1</i> |
| | pedido = null | <i>No</i> | <i>2</i> |

| Batería de Pruebas - Escenario Base | | | |
|-------------------------------------|------------------|--|--|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | Pedido | ("a","111","a a"),1,false,false,10,Constantes. ZONA_SIN_ASFALTAR | No hay vehiculos disponibles |
| Batería de Pruebas - Escenario 2 | | | |
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 2 | Pedido | {("facundo","123","Facundo"),3,true,true,10,PELIGROSA), ("thiago","321","Thiago"),1,false,false,3,STANDARD),("nehuen","4567","nehuen"),8,false,true,1,PELIGROSA)} | Existen vehiculos que satisfagan esos pedidos |
| Batería de Pruebas - Escenario 5 | | | |
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 3 | Pedido | ("thiago","321","Thiago"),1,true,false,3,STANDARD) | "Hay vehiculos que estan disponibles" |
| 4 | Pedido | ("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | "No hay vehiculos cargados que satisfagan este pedido" |
| 5 | Pedido | ("sofi","4444","Sofi"),8,true,true,1,PELIGROSA) | "Hay vehiculos que satisfacen este pedido" |
| 6 | Pedido | ("sofi","4444","Sofi"),5,true,false,20,"ZONA_STANDARD") | "No hay vehiculos que satisfacen este pedido" |
| 7 | Pedido | ("sofi","4444","Sofi"),10,false,false,20,"ZONA_STANDARD") | "No hay vehiculos que satisfacen este pedido" |

- **agregarVehiculo(Vehiculo vehiculo):** Al querer cargar un vehiculo repetido, la excepcion VehiculoRepetidoException está mal construida (e.getVehiculoExistente() = null)

| Tabla de Particiones | | | |
|----------------------|--|-----------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| Vehiculo | vehiculo!=null, Moto, patente valida no existente en la coleccion | Si | 1 |
| | vehiculo!=null, Moto, patente valida existente en la coleccion | Si | 2 |
| | vehiculo!=null, Auto, patente valida no existente en la coleccion | Si | 3 |
| | vehiculo!=null, Auto, patente valida existente en la coleccion | Si | 4 |
| | vehiculo!=null, Combi, patente valida no existente en la coleccion | Si | 5 |
| | vehiculo!=null, Auto, patente valida existente en la coleccion | Si | 6 |
| | vehiculo== null | No | 7 |

| Batería de Pruebas - Escenario 2 | | | |
|----------------------------------|------------------|-----------------------------|---------------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | Vehiculo | new Moto("XYZ789") | Vehiculo agregado correctamente |
| 2 | Vehiculo | new Moto("pat333") | VehiculoRepetidoException |
| 3 | Vehiculo | new Auto("ASD789",4,true) | Vehiculo agregado correctamente |
| 4 | Vehiculo | new Auto("abc123",4,true) | VehiculoRepetidoException |
| 5 | Vehiculo | new Combi("DDD789",6,false) | Vehiculo agregado correctamente |

| | | | |
|---|----------|---|----------------------------------|
| 6 | Vehiculo | <i>new</i> <i>Combi("combi222",10,false)</i> | <i>VehiculoRepetidoException</i> |
|---|----------|---|----------------------------------|

- **calificacionDeChofer(Chofer chofer):** Cuando se quiere obtener la calificacion de un chofer que no tuvo viajes el metodo deberia arrojar SinViajeException pero no lo hace.

| Tabla de Particiones | | | |
|----------------------|--|-----------------------------------|--|
| Dato de entrada | Descripción de la Clases de equivalencia | Aplica ? (cumple el contrato?) | Identificador de clase de equivalencia |
| chofer | <i>chofer != null</i> | <i>si</i> | 1 |
| | <i>chofer == null</i> | <i>no</i> | 2 |

| Batería de Pruebas - Escenario 3 | | | |
|----------------------------------|------------------|---|--------------------------|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | <i>chofer</i> | <i>ChoferPermanente("1234567", "Roberto", 2020, 0); con una calificación de 5</i> | 5.0 |
| 2 | <i>chofer</i> | <i>ChoferPermanente("1234567", "Roberto", 2020, 0); con varias calificaciones</i> | |
| 3 | <i>chofer</i> | <i>ChoferPermanente("1234568", "Alberto", 2019, 3) Sin viajes iniciados</i> | <i>SinViajeException</i> |

Controlador (Test de Integración):

Dentro del repositorio de [GitHub](#) se encuentra un archivo con el diagrama para generar los casos de prueba del test de integración.

- **nuevoChofer():** Se encontró que al querer registrar un chofer repetido no se envia ningun mensaje (envía null)

| Batería de Pruebas | | | |
|--------------------|------------------|-------|-----------------|
| Número | Datos de entrada | Valor | Salida Esperada |

| de prueba | | | |
|---|-------------------------|------------|--|
| 1 | vista.getTipoChofer() | TEMPORARIO | El metodo registra al Chofer Temporario obteniendo los datos de la vista y lo persiste |
| | vista.getNombreChofer() | "Alberto" | |
| | vista.getDNIChofer() | "1234567" | |
| | vista.getAnioChofer() | N/A | |
| | vista.getHijosChofer() | N/A | |
| 2 | vista.getTipoChofer() | PERMANENTE | El metodo registra al Chofer Permanente obteniendo los datos de la vista y lo persiste |
| | vista.getNombreChofer() | "Alberto" | |
| | vista.getDNIChofer() | "1234567" | |
| | vista.getAnioChofer() | 2020 | |
| | vista.getHijosChofer() | 1 | |
| Batería de Pruebas - Escenario 2 | | | |
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 4 (se registra un chofer ya existente) | vista.getTipoChofer() | TEMPORARIO | El metodo obtiene los datos correctamente. El mensaje de error es CHOFER_YA_REGISTRADO |
| | vista.getNombreChofer() | "Alberto" | |
| | vista.getDNIChofer() | "1234567" | |
| | vista.getAnioChofer() | 2020 | |
| | vista.getHijosChofer() | 1 | |

- **nuevoViaje():** Al querer generar un nuevo viaje a un cliente que ya tiene un viaje en curso no se envia bien el mensaje CLIENTE_CON_VIAJE_PENDIENTE en cambio se envia que el pedido no se encuentra en la lista. Cuando se comprobó que efectivamente el pedido se agrega. También al querer iniciar un viaje con un chofer no disponible el sistema arroja el mismo mensaje "El Pedido no figura en la lista".

| Batería de Pruebas - Escenario 2 | | | |
|---|---|--|--|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | vista.getPedidoSeleccionado() | ((("facundo","123","Facundo"),3,true,true,10,PELIGROSA)) | El metodo registra el nuevo Viaje obteniendo los datos de la vista y lo persiste |
| | vista.getChoferDisponibleSeleccionado() | ("1234567","Roberto",2020,0) | |

| | | | |
|--|--|---|---|
| | vista.getVehiculoDisponibileSeleccionado() | ("abc123",4,true) | |
| 2 | vista.getPedidoSeleccionado() | ("facundo","123","Facundo"),1,false,false,2,Constantes.ZONA_STANDARD) | El metodo arroja el error PEDIDO_INEXISTENTE |
| | vista.getChoferDisponibileSeleccionado() | ("1234567","Roberto",2020,0) | |
| | vista.getVehiculoDisponibileSeleccionado() | ("abc123",4,true) | |
| 3 (se agrega a la coleccion de pedidos el pedido seleccionado) | vista.getPedidoSeleccionado() | ("facundo","123","Facundo"),1,false,false,2,Constantes.ZONA_STANDARD) | El metodo arroja el error CLIENTE_CON_VIAJE_PENDIENTE |
| | vista.getChoferDisponibileSeleccionado() | ("11111111","Javier") | |
| | vista.getVehiculoDisponibileSeleccionado() | ("pat333") | |
| 4 | vista.getPedidoSeleccionado() | ((("facundo","123","Facundo"),3,true,true,10,PELIGROSA) | El metodo arroja el error VEHICULO_NO_VALIDO |
| | vista.getChoferDisponibileSeleccionado() | ("1234567","Roberto",2020,0) | |
| | vista.getVehiculoDisponibileSeleccionado() | ("pat333") | |
| Batería de Pruebas - Escenario 3 | | | |
| 5 | vista.getPedidoSeleccionado() | ("thiago","321","Thiago"),1,false,false,3,STANDARD) | El metodo arroja el error VEHICULO_NO_DISPONIBLE |
| | vista.getChoferDisponibileSeleccionado() | ("1234568","Alberto",2019,3) | |
| | vista.getVehiculoDisponibileSeleccionado() | ("abc123",4,true) | |
| 6 | vista.getPedidoSeleccionado() | ("thiago","321","Thiago"),1,false,false,3,STANDARD) | El método arroja el error CHOFER_NO_DISPONIBLE |
| | vista.getChoferDisponibileSeleccionado() | ("1234567","Roberto",2020,0) | |
| | vista.getVehiculoDisponibileSeleccionado() | ("pat333") | |

- **nuevoPedido():** Al querer generar un pedido válido se asigna mal el atributo de km del pedido.

| Batería de Pruebas - Escenario 6 | | | |
|---|---|--|---|
| Número de prueba | Datos de entrada | Valor | Salida Esperada |
| 1 | Empresa.getInstance().getUsuarioLogeado() | ("thiago", "321", "Thiago") | El sistema no agrega el pedido al cliente y se arroja el error SIN_VEHICULO_PARA_PEDIDO |
| | vista.getPedidoSeleccionado() | ((("thiago", "321", "Thiago"), 3, true, true, 5, ZONA_STANDARD)) | |
| 2 | Empresa.getInstance().getUsuarioLogeado() | ("facundo", "123", "Facundo") | El sistema crea el viaje solicitado por el cliente, asigna el vehiculo y el chofer, registra el pago y la calificacion por parte del Cliente exitosamente |
| | vista.getPedidoSeleccionado() | ((("facundo", "123", "Facundo"), 3, false, false, 5, ZONA_STANDARD)) | |
| | vista.getChoferDisponibleSeleccionado() | ("1234567", "Roberto", 2020, 0) | |
| | vista.getVehiculoDisponibleSeleccionado() | ("dfg456", 3, false) | |
| 3 | Empresa.getInstance().getUsuarioLogeado() | ("nehuen", "4567", "Nehuen") | El sistema rechaza la calificacion sin un viaje y muestra el mensaje correspondiente a la excepcion ClienteSinViajePerdienteException |
| | vista.getPedidoSeleccionado() | ((("nehuen", "4567", "Nehuen"), 3, false, false, 5, ZONA_STANDARD)) | |

Test de Persistencia:

Al realizar el test de persistencia no se encontró ningún error.

Test de GUI:

- **Test Mensajes:**
 - **testRegistroChoferRepetido():** al registrar un nuevo chofer en la ventana de administrador, no salta el mensaje de chofer repetido al registrar al mismo chofer, es decir, puedo apretar el botón de “aceptar

chofer" múltiples veces y registra a todos los choferes sin que salte el mensaje "**CHOFER_YA_REGISTRADO**"

-Test EnabledDisabled:

- testAdminAgregaVehiculoAutoPlazaInvalida_0():

al registrar un vehículo de tipo Auto con "0" cantidad de plazas, se habilita el botón de "ACEPTAR VEHICULO" cuando el rango de cantidad de plazas válido para que se habilite el botón es de 1 a 4.

-Test verif_JtextField:

-testregistrovehiculo_correctamente():

al registrar un vehículo no se borran los JtextField correspondientes a cantidad de plazas y Patente.

-testregistrochofer_correctamente():

al registrar un chofer no se borran los JtextField correspondientes a DNI,nombre,cantidad de hijos y año ingreso.

Conclusiones

La experiencia de testing ha sido desafiante, requiriendo un análisis exhaustivo del código y la documentación para diseñar casos de prueba efectivos. Se aprendió la importancia de la planificación y la documentación de las pruebas, así como el uso de herramientas como JUnit. El proceso permitió identificar errores críticos en diferentes componentes del sistema, lo que contribuirá a mejorar su calidad y robustez. La dificultad mayor residió en la identificación de los escenarios de prueba más relevantes. Se destaca la importancia de las pruebas de integración para detectar problemas que no se evidencian en las pruebas unitarias.