Proyecto 3

Ana María Sánchez Mejía –202013587

David Leonardo Manrique Lesmes –201913129

Maria Paula Nizo Vega – 202213902

Tabla de contenido

Contexto del Problema

Se realizará una aplicación que permita al usuario poder llevar a cabo de forma sencilla una reserva de un automóvil en la ciudad, asimismo debe posibilitar que los administradores o empleados de la empresa de alquiler manejen la información y tengan acceso a las modificaciones que dependerán de sus responsabilidades. A continuación, se presentan las principales características del contexto y sus actuales funciones agregadas:

Lista funciones:

Por consola:

1. ActualizarEstadoVehiculo
2. RegistrarEliminarVehiculo$2
3. Registrar EliminarVehiculo$3
4. Registrar EliminarVehiculo$4
5. Registrar EliminarVehiculo$5
6. RegistrarEliminar Vehiculo$6
7. Registrar EliminarVehiculo$7
8. RegistrarEliminarVehiculo
9. RegistrarEntregaVehiculo$1

10. RegistrarEntregaVehiculo$2
11. RegistrarEntregaVehiculo
12. RegistrarUsuario$1
13. RegistrarUsuario$2
14. RegistrarUsuario$3
15. RegistrarUsuario
16. SeguroExtra$1
17. SeguroExtra$2
18. SeguroExtra
19. VehiculoMantenimiento$1
20. VehiculoMantenimiento
21. Ventana
22. VentanaClientes
23. VentanaEmpleados
24. PanelOpcionesAdminSede
25. PanelOpcionesCliente
26. PanelOpcionesLimpiador
27. PanelOpciones Recepcion
28. PanelPrincipalClientes$1
29. PanelPrincipalClientes
30. Panel PrincipalEmpleados$1
31. PanelPrincipalEmpleados
32. PanelRegistroNuevoUsuario$1
33. PanelRegistroNuevoUsuario
34. PruebaMapa$1
35. PruebaMapa$2
36. PruebaMapa
37. RegistrarCategoria$1
38. RegistrarCategoria$2
39. RegistrarCategoria
40. Registrar DevolucionVehiculo$1
41. Registrar DevolucionVehiculo$2
42. Registrar DevolucionVehiculo
43. RegistrarEliminarSede$1
44. RegistrarEliminarSede$2
45. Registrar EliminarSede$3
46. RegistrarEliminarSede
47. RegistrarEliminarSeguro$1
48. RegistrarEliminar Seguro$2
49. Registrar EliminarSeguro$3
50. Registrar EliminarSeguro
51. RegistrarEliminarVehiculo$1
52. DisponibilidadSedes
53. EditarFechas$1
54. EditarFechas
55. EditarReserva$1
56. EditarReserva
57. EditarSedes$1

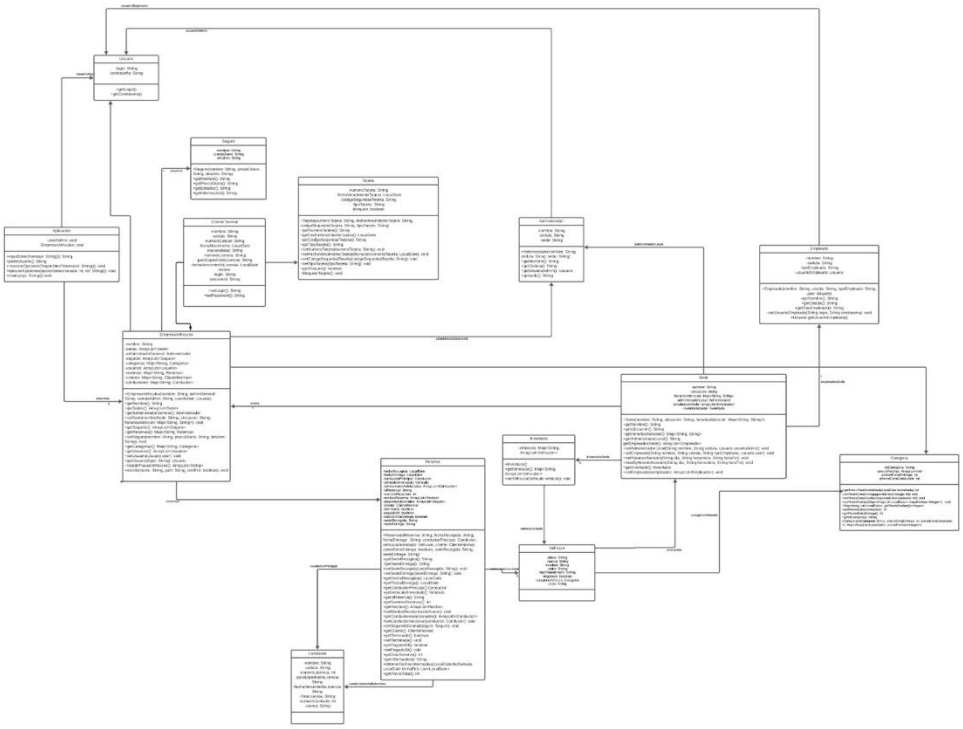
58. EditarSedes$2
59. EditarSedes$3
60. EditarSedes$4
61. EditarSedes
62. EliminarUsuario$1
63. EliminarUsuario$2
64. EliminarUsuario
65. Generar Traslado$1
66. Generar Traslado$2
67. Generar Traslado$3
68. Generar Traslado
69. Graficador
70. ModificarPreciosCategoria$1
71. ModificarPreciosCategoria$2
72. Modificar PreciosCategoria$3
73. Modificar PreciosCategoria
74. OcupacionSede$1
75. OcupacionSede$2
76. OcupacionSede
77. PanelFondoGeneral
78. PanelOpcionesAdminGeneral
79. ActualizarEstadoVehiculo$1
80. ActualizarEstadoVehiculo
81. ArchivoLog$1
82. ArchivoLog$2
83. ArchivoLog
84. ConductorExtra$1
85. ConductorExtra
86. CrearReserva$1
87. CrearReserva$2
88. CrearReserva$3
89. CrearReserva$4
90. CrearReserva
91. Disponibilidad Sedes$1
92. Disponibilidad Sedes$2

Por modelo:

1. Administrador
2. Categoria
3. ClienteEspecial
4. ClienteNormal
5. Conductor
6. Empleado
7. EmpresaVehiculos
8. Inventario
9. Reserva

10. Sede
11. Seguro
12. Tarjeta
13. TrasladoReserva
14. Usuario
15. Vehiculo
16. Otros
17. Pasarela
18. PayPal
19. PayU
20. Sire

Diagrama de clases UML



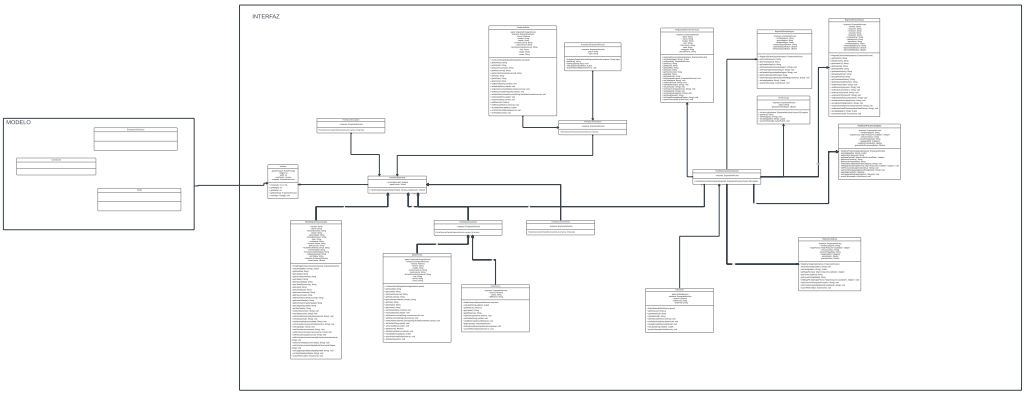


Diagrama de alto nivel

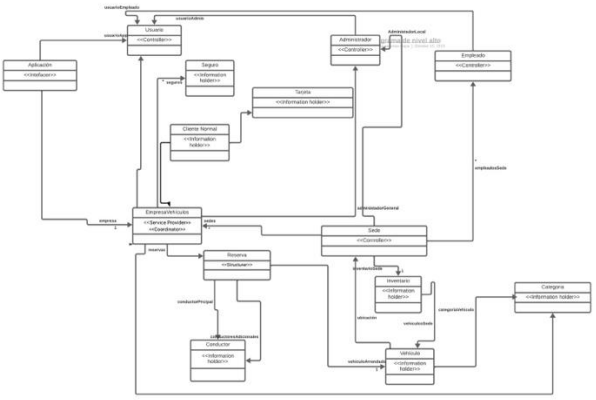


Diagrama de clases de secuencia

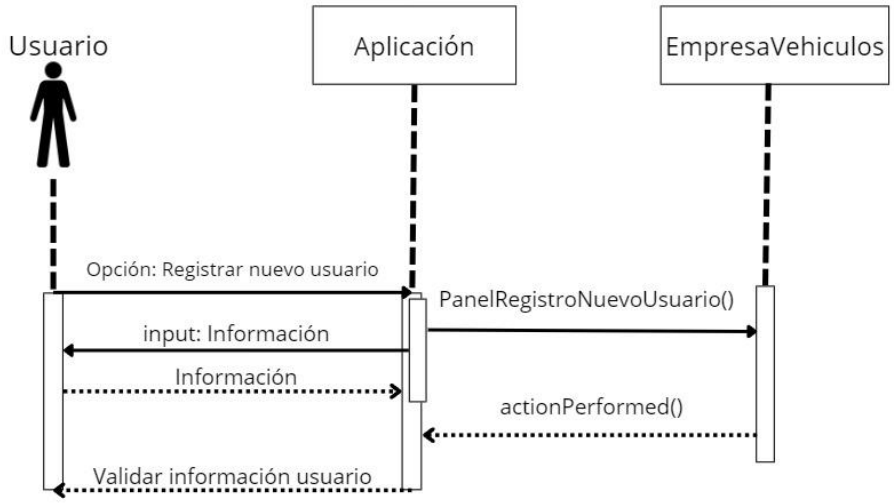


Figura 1. Crear usuario

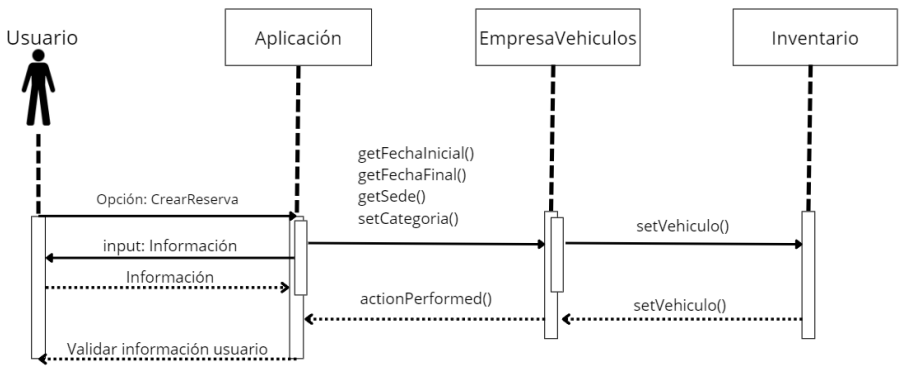


Figura 2. Crear Reserva

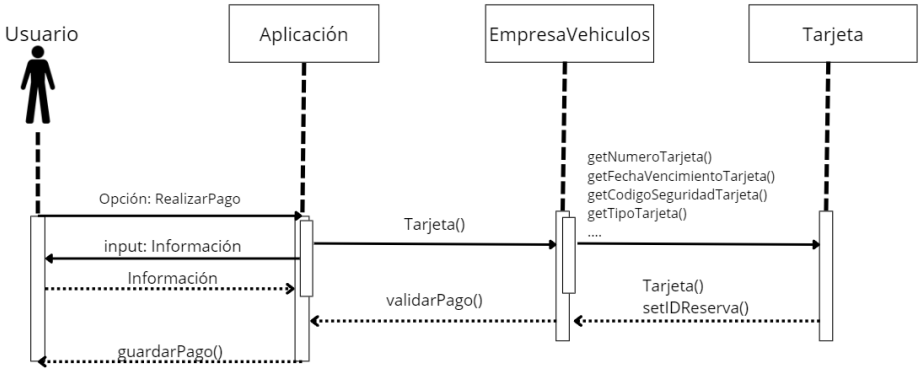


Figura 2. Realizar Pago

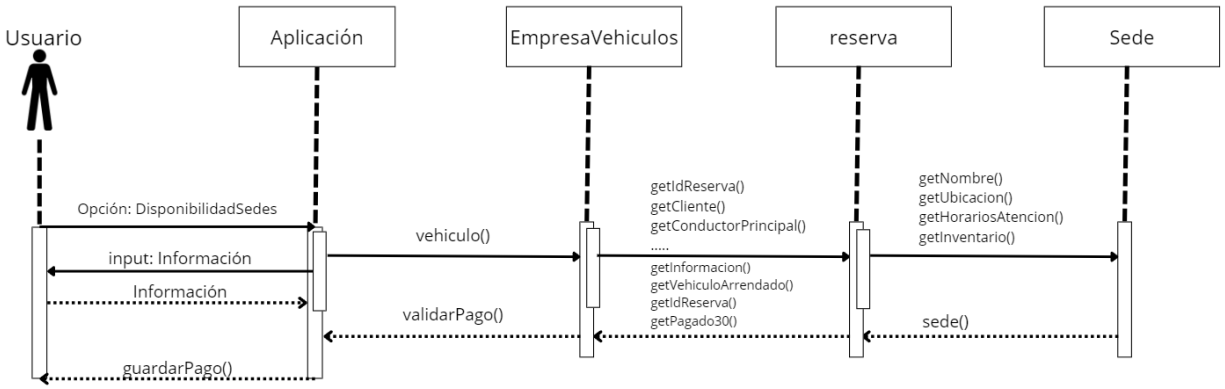


Figura 3. Sede Disponible

Métodos de algunas de las clases

Por consola:

- PanelFondoGeneral:
  - PanelFondoGeneral(ventanaPrincipal: Ventana, panelCentro: JPanel)
- PanelOpcionesAdminGeneral:
  - PanelOpcionesAdminGeneral(empresa: EmpresaVehiculos) throws IOException
- PanelPrincipal:
  - PanelPrincipal(ventanaPrincipal: Ventana)
  - setUsuario(user: String): void
  - setContrasena(contrasena: String): void
  - actionPerformed(pEvento: ActionEvent): void

- PanelRegistroNuevoUsuario:
    - PanelRegistroNuevoUsuario(empresa: EmpresaVehiculos)
    - setLabel(palabra: String): JLabel
    - Métodos de acceso y modificación para varios atributos.
- RegistrarCategoria:
    - RegistrarCategoria(empresa: EmpresaVehiculos)
    - setNombreCategoria(text: String): void
    - setLabel(palabra: String): JLabel
    - Métodos de acceso y modificación para varios atributos.
- RegistrarEliminarSede:
    - RegistrarEliminarSede(empresa: EmpresaVehiculos)
    - Métodos de acceso y modificación para varios atributos.
    - setLabel(palabra: String): JLabel
    - actionPerformed(e: ActionEvent): void
- RegistrarEliminarSeguro:
    - RegistrarEliminarSeguro(empresa: EmpresaVehiculos)
    - Métodos de acceso y modificación para varios atributos.
    - setLabel(palabra: String): JLabel
    - actionPerformed(e: ActionEvent): void
- Ventana:
    - panelPrincipal: PanelPrincipal
    - height: int
    - width: int
    - userAdmin: Usuario
    - empresa: EmpresaVehiculos
    - Ventana(h: int, w: int)
    - getHeight(): int
    - getWidth(): int
    - getEmpresa(): EmpresaVehiculos
    - main (args: String[]): void
- ModificarPreciosCategoria:
    - ModificarPreciosCategoria(empresa: EmpresaVehiculos)
    - setLabel(palabra: String)
    - GetNombreCategoria(): String
    - GetPrecioCondAux(): String
    - GetPrecioEntregaSede(): String
- GenerarTraslado:
    - GenerarTraslado(empresa: EmpresaVehiculos)
    - getPlacaTraslado(): String
    - getEmpleado(): String
    - getFecha(): String
    - getSedeDestino(): String
    - setPlacaTraslado(placaTraslado: String)
    - setEmpleado(empleado: String)
- SeguroExtra:
    - SeguroExtra(RegistrarEntregaVehiculo panel)

    - getReserva(): Reserva
    - getSeguro(): String
    - setReserva(reserva: Reserva)
    - setSeguro(seguro: String)
- Pasarela:

Método abstracto guardarPago:

Firma: public abstract void guardarPago(Tarjeta tarjeta, String idReserva, String porcentaje, String total) throws IOException;

Descripción: Este método representa la responsabilidad de guardar la información de un pago. Toma como parámetros una tarjeta (Tarjeta), un identificador de reserva (idReserva), un porcentaje y un total. Puede lanzar una excepción de tipo IOException.

Método abstracto validarPago:

Firma: public abstract boolean validarPago(Tarjeta tarjeta);
Descripción: Este método representa la responsabilidad de validar un pago. Toma como parámetro una tarjeta (Tarjeta) y devuelve un valor booleano que indica si el pago es válido o no.

- Otros

Almacenar información sobre un vehículo "Otros":

Atributos: Id, color, tipo, porcentaje, sede.
Constructor: public Otros(String color, String Id, String tipo, float porcentaje, String sede) para inicializar los atributos al crear una instancia de la clase.
Proporcionar datos sobre un vehículo "Otros":

Métodos getters:
getColor(): Retorna el color del vehículo.
getId(): Retorna el identificador del vehículo.
getTipo(): Retorna el tipo del vehículo.
getPorcentaje(): Retorna el porcentaje asociado al vehículo.
getSede(): Retorna la sede asociada al vehículo.
Permitir la modificación de información del vehículo "Otros":

Métodos setters:
setColor(String color): Establece el color del vehículo.
setId(String id): Establece el identificador del vehículo.
setTipo(String tipo): Establece el tipo del vehículo.
setPorcentaje(float porcentaje): Establece el porcentaje asociado al vehículo.
setSede(String sede): Establece la sede asociada al vehículo.

Decisiones de Diseño

- Se tomó la decisión de que la clase Aplicación sea un <> para las opciones del menú de los usuarios. Esta clase tiene la función de ejecutar los procesos de autenticación, inicio de sesión, registro en la plataforma y un menú de opciones para la categoría del usuario registrado.
- Se tomó la decisión de que la clase Sede almacene un inventario de los carros que tiene cada una de las sedes.
- Se tomó la decisión de definir la clase Aplicación como un <> ya que, a través de la interacción por consola de los menús, se delega a las otras clases que están dentro del paquete de la lógica que realicen sus funciones.
- Se tiene en cuenta que para cerrar la sesión de la aplicación se deben cerrar por medio de la x de las ventanas después de hacer los arreglos que cada usuario desee y al cerrarlo se guardaran los cambios.
- Para poder acceder a la gráfica de ocupación el único que la puede visualizar va a ser el administrador general
- Se tomó la decisión de crear una clase para otros transportes que se llama vehículo y todo lo que se tenga que tramitar con los carros se llama carro
- Una de las nuevas clases se llama otros en esta clase se hace el recuentro de los otros tipos de vehículos.

Roles

Asignar roles a las clases es una parte importante del diseño de un sistema. Los roles definen cómo cada clase se relaciona con otras clases y qué responsabilidades tiene en el sistema.

- Control de contraseñas – Usuario
- Control de usuarios – Usuario
- Carga y lectura de archivos de datos – Empresa Vehículos
- Representa a un usuario del sistema. – Usuario
- Almacenamiento de la información de vehículos – Inventario
- Realiza una reserva de un vehículo – Reserva
- Almacenamiento de la información de un nuevo vehículo – Vehículo-Carro
- Almacenamiento de la información de la venta - Pasarela
- Almacenamiento de la información de una tarjeta – Tarjeta
- Almacenamiento de la información de los seguros – Seguro
- Almacenamiento de la información de los inventarios – Sede
- Almacenamiento de la información de los clientes – Clientes

Responsabilidades

Carro:

Responsabilidades:

Almacenar y proporcionar información detallada sobre el vehículo, incluyendo placa, marca, modelo, y más. Mantener un registro del estado de alquiler del vehículo. Proporcionar datos esenciales para la gestión de flotas de vehículos.

Usuario:

Responsabilidades:

Almacenar credenciales de acceso, incluyendo nombre de usuario y contraseña. Identificar el rol del usuario en el sistema (administrador, empleado, cliente). Facilitar la autenticación y el control de acceso.

TrasladoReserva:

Responsabilidades:

Almacenar información relevante sobre el traslado, como placa del vehículo, fecha, y sedes de salida y llegada. Facilitar la administración de traslados relacionados con reservas.

Tarjeta:

Responsabilidades:

Almacenar detalles de la tarjeta, incluyendo número, fecha de vencimiento y tipo. Gestionar el estado de bloqueo de la tarjeta para pagos seguros.

Seguro:

Responsabilidades:

Almacenar información sobre el seguro, incluyendo nombre, precio y detalles adicionales. Proporcionar datos necesarios para la cobertura de seguros.

Sede:

Responsabilidades:

Almacenar información de la sede, como ubicación, horarios de atención y personal. Facilitar la gestión y operación de las distintas sedes.

Reserva:

Responsabilidades:

Almacenar información completa sobre la reserva, incluyendo fechas, vehículos, conductores y detalles de cobro. Calcular el precio total de la reserva en función de los elementos seleccionados.

Recibo:

Responsabilidades:

Almacenar información sobre el recibo, incluyendo el ID de la reserva, tipo de objeto y precio. Registrar transacciones y pagos relacionados con reservas.

Inventario:

Responsabilidades:

Almacenar información detallada sobre vehículos clasificados por categoría. Permitir la adición y gestión de vehículos en el inventario.

EmpresaVehiculo:

Responsabilidades:

Coordinar y gestionar sedes, seguros, usuarios, reservas y más. Facilitar el acceso y la gestión de todos los componentes del sistema. Realizar funciones de autenticación y control de acceso en el sistema.

Empleado:

Responsabilidades:

Almacenar información personal y roles de los empleados. Proporcionar acceso a detalles individuales de los empleados. Facilitar la creación y gestión de usuarios asociados a los empleados.

Conductor, ClienteNormal, ClienteEspecial:

Responsabilidades:

Almacenar detalles específicos de conductores y clientes, como licencias, contactos y datos personales. Proporcionar acceso a los atributos individuales de conductores y clientes. Contribuir a la gestión de conductores y clientes en el sistema.

Otros:

Responsabilidades:

la clase Otros es responsable de representar y gestionar la información de un vehículo del tipo "Otros", proporcionando métodos para acceder y modificar sus atributos. Estas responsabilidades siguen principios de encapsulamiento al ocultar la implementación interna y proporcionar interfaces bien definidas para interactuar con los objetos de la clase.

Pasarela:

La clase Pasarela es abstracta, lo que significa que no se pueden crear instancias directas de esta clase. Se espera que las clases que hereden de Pasarela proporcionen implementaciones concretas para estos métodos abstractos. Esta estructura permite definir un contrato básico para las pasarelas de pago, pero deja la implementación específica de cómo se almacenan y validan los pagos a las clases concretas que extienden Pasarela. Esto sigue el principio de abstracción y permite una mayor flexibilidad y extensibilidad en el diseño del sistema.