Sistema de Gestion de Alquiler de Vehiculos

Alex Travez, Mateo Oviedo

Escuela Politecnica Nacional Proyecto 2 – Programacion Avanzada Curso: Gr1

Fecha: 1/02/2025

***Abstract*—Este informe describe el desarrollo de un sistema interactivo en consola para la gestion de clientes, vehiculos y reservas de una empresa de alquiler de automoviles. La solucion utiliza Programacion Orientada a Objetos (POO), serializacion para persistencia de datos y manejo de excepciones. Se detalla la funcionalidad de las principales clases, metodos implementados y la tecnologia aplicada.**

1. Introduccion

El presente informe documenta el desarrollo del *Sistema de Gestion de Alquiler de Vehiculos* para MiAlquiler.co. El objetivo principal del proyecto es implementar una aplicacion interactiva que permita gestionar clientes, vehiculos y reser- vas. Este sistema ha sido disen˜ado aplicando principios de Programacion Orientada a Objetos (POO), serializacion para persistencia de datos y manejo robusto de excepciones.

1. Clases y Su Funcionalidad
2. *Cliente*

**Descripcion:** Representa a los usuarios del sistema.

# Atributos:

* + nombre (String): Nombre del cliente.
  + identificacion (String): Identificador del cliente.
  + contacto (String): Informacion de contacto del cliente.

**Funcionalidad:** Permite almacenar y gestionar los datos personales de los clientes.

1. *Vehiculo*

**Descripcion:** Modela los vehiculos disponibles para alquiler.

# Atributos:

* + marca (String): Marca del vehiculo.
  + modelo (String): Modelo del vehiculo.
  + tipo (String): Tipo del vehiculo (auto, camioneta, mo- tocicleta).
  + disponible (boolean): Indica si el vehiculo esta disponible.

**Funcionalidad:** Controla la disponibilidad y detalles tecni- cos de los vehiculos.

1. *Reserva*

**Descripcion:** Asocia un cliente con un vehiculo en un periodo especifico.

# Atributos:

* + cliente (Cliente): Cliente que realiza la reserva.
  + vehiculo (Vehiculo): Vehiculo reservado.
  + fechaInicio (Date): Fecha de inicio del alquiler.
  + fechaFin (Date): Fecha de fin del alquiler.

**Funcionalidad:** Gestiona las fechas y la relacion entre cliente y vehiculo.

1. *AgenciaAlquiler*

**Descripcion:** Clase principal que coordina todas las opera- ciones.

# Atributos:

* + clientes (List*<*Cliente*>*): Lista de clientes registra- dos.
  + vehiculos (List*<*Vehiculo*>*): Lista de vehiculos disponibles.
  + reservas (List*<*Reserva*>*): Historial de reservas.

# Funcionalidad:

* + Agrega o elimina clientes y vehiculos.
  + Realiza reservas y valida la disponibilidad.

1. *Serializador*

**Descripci’on:** Maneja la persistencia de datos.

# Metodos:

* + serializar(String archivo, Object objeto): Guarda objetos en archivos binarios.
  + deserializar(String archivo): Recupera datos desde archivos binarios.

**Funcionalidad:** Permite mantener la informacion entre se- siones del sistema.

1. *Excepciones Personalizadas*
   * DatosInvalidosException: Lanza errores ante en- tradas incorrectas, como formatos de fecha invalidos.
   * VehiculoOcupadoException: Lanza errores cuando un vehiculo ya esta reservado.
2. Variables y Su Proposito
   * nombre (String): Almacena el nombre del cliente en la clase Cliente.
   * identificacion (String): Identificador del cliente en la clase Cliente.
   * disponible (boolean): Indica la disponibilidad del vehiculo en la clase Vehiculo.
   * fechaInicio (Date): Fecha de inicio del alquiler en la clase Reserva.
   * clientes (List*<*Cliente*>*): Lista de clientes registra-

dos en AgenciaAlquiler.

* + reservas (List*<*Reserva*>*): Lista de reservas activas en AgenciaAlquiler.

1. Metodos y Funciones Clave
   * agregarCliente(Cliente cliente): Agrega un nuevo cliente a la lista en AgenciaAlquiler. Valida que la identificaci’on no este duplicada.
   * realizarReserva(Reserva reserva): Crea una reserva si el vehiculo esta disponible. Lanza VehiculoOcupadoException si el vehiculo ya esta reservado.
   * serializar(String archivo, Object objeto): Convierte objetos a formato binario y los guarda en un archivo.
   * cargarDatos(): Recupera datos utilizando el

Serializador al iniciar el programa.

* + verificarDisponibilidad(Vehiculo vehiculo): Retorna true si el vehiculo puede ser alquilado.
  + mostrarVehiculosDisponibles(): Lista todos los vehiculos con disponible = true.

1. Tecnologias y Principios Aplicados
2. *Programacion Orientada a Objetos (POO)*
   * **Abstraccion**: Clases como Cliente y Vehiculo

modelan entidades del mundo real.

* + **Encapsulamiento**: Atributos privados con metodos pub- licos para acceder y modificar datos.
  + **Herencia**: Excepciones personalizadas heredan de la clase Exception.

1. *Serializacion*

Permite almacenar datos en archivos binarios para garantizar la persistencia de la informacion.

1. *Manejo de Excepciones*

Mejora la robustez del sistema al gestionar entradas invali- das y situaciones no permitidas.

1. Conclusion

El *Sistema de Gestion de Alquiler de Vehiculos* cumple con los requisitos funcionales definidos por MiAlquiler.co, permi- tiendo la gestion eficiente de clientes, vehiculos y reservas. La implementacion de POO garantiza un codigo modular y escal- able, mientras que la serializacion asegura la persistencia de

datos. El manejo robusto de excepciones mejora la experiencia del usuario al prevenir errores criticos.

1. Recomendaciones

* En lugar de usar archivos binarios para la persistencia, se podria integrar una base de datos SQL o NoSQL para manejar grandes volumenes de datos de manera mas eficiente.
* Agregar mas validaciones, como la verificacion de for- matos de correo electronico o numeros de telefono, para garantizar la integridad de los datos.

1. Bibliografia

* Bloch, J. (2017). *Effective Java*. Addison-Wesley.
* PlantUML. (2023). PlantUML Documentation. Recuper- ado de <https://plantuml.com/>
* Analisis y clases de diseño, Lenguaje UML (Material de clase, presentado por David Mejia)[Consultado: 1-January-2025].

1. Preguntas
2. *1-Desafio mas grande afrontado?*

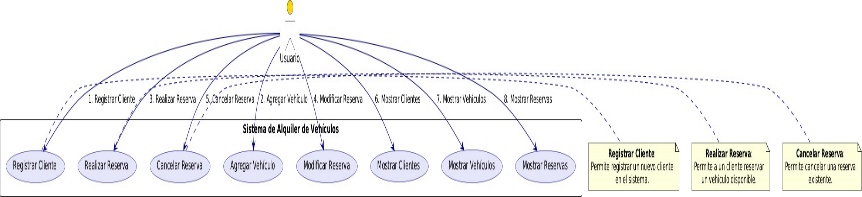
El desafio mas grande que afrontamos durante el desarrollo del proyecto fue la implementacion de la serializacion y el manejo de excepciones. La serializacion de datos para garan- tizar la persistencia de la informacion (clientes, vehiculos y reservas) requirio un enfoque cuidadoso para asegurar que los datos se almacenaran y recuperaran correctamente. Ademas, el manejo de excepciones fue crucial para garantizar que el sistema fuera robusto y manejara adecuadamente entradas invalidas, como fechas incorrectas o vehiculos no disponibles. Este desafio nos obligo a profundizar en conceptos avanzados de Java y a disen˜ar un sistema que fuera tanto funcional como seguro.

1. *¿Que nota merece?*

Consideramos que el proyecto merece una nota de 8/10. Hemos cumplido con todos los requisitos funcionales y no funcionales, implementando un sistema completo que incluye gestion de clientes, vehiculos y reservas, junto con la serializa- cion de datos y el manejo de excepciones. Ademas, aplicamos los principios de la programacion orientada a objetos (abstrac- cion, encapsulamiento, herencia y polimorfismo) de manera efectiva.

1. *Experiencia de aprendizaje*

La experiencia de aprendizaje durante este proyecto fue ex- tremadamente enriquecedora. Aprendimos a aplicar conceptos teoricos de programacion orientada a objetos en un proyecto real, lo que nos permitio comprender mejor c’omo dise nar sistemas escalables y mantenibles. Ademas, enfrentarnos a desafios como la serializacion y el manejo de excepciones nos ayudo a desarrollar habilidades de depuracion y resolucion de problemas. Tambien mejoramos nuestra capacidad para trabajar en equipo, dividir tareas y coordinar esfuerzos para alcanzar un objetivo comun. En general, este proyecto nos permitio consolidar nuestros conocimientos de Java y nos preparo para enfrentar proyectos mas complejos en el futuro.

*Diagramas del Sistema*

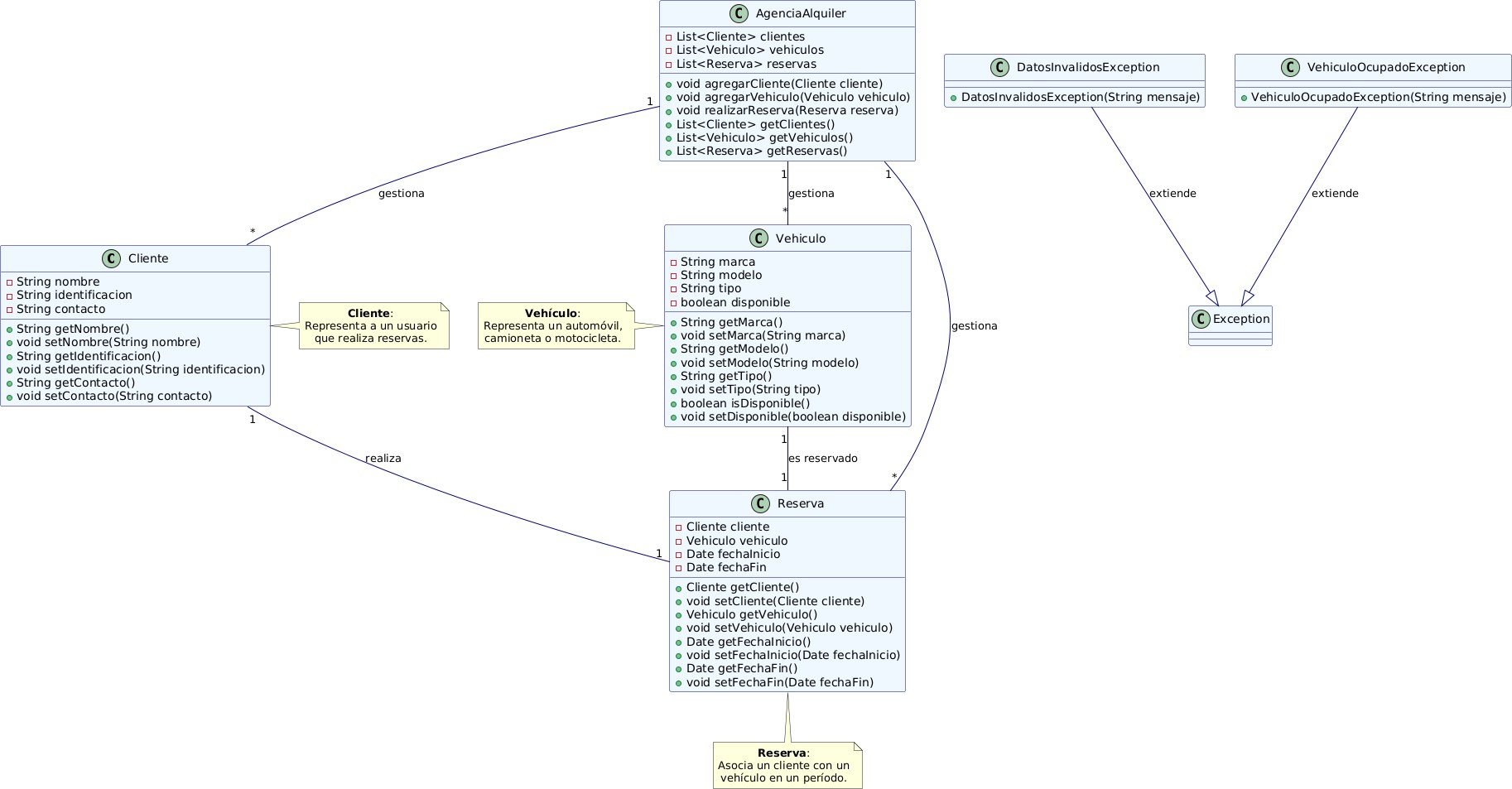
Fig. 1: Diagrama de Casos de Uso

Fig. 2: Diagrama de Clases

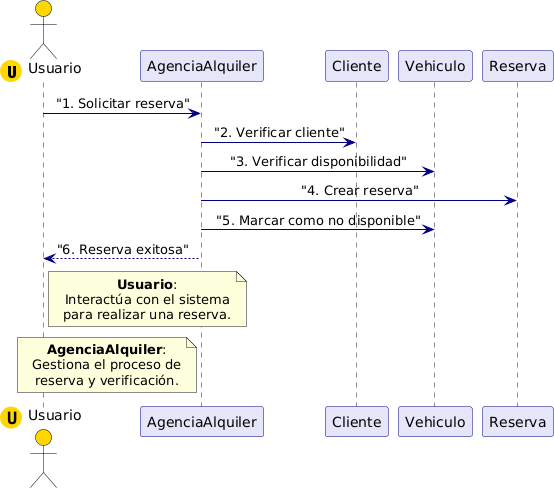


Fig. 3: Diagrama de Secuencia

Anexos

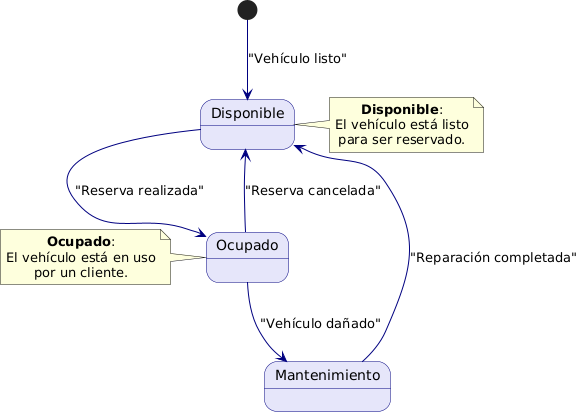


Fig. 4: Diagrama de Estados

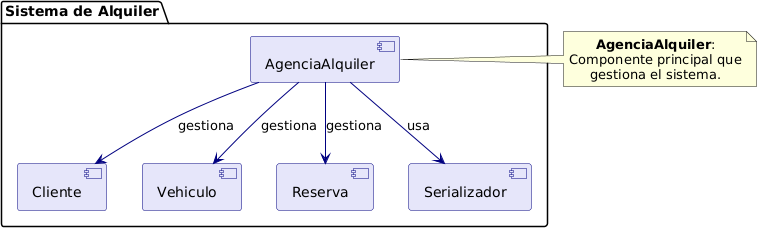


Fig. 5: Diagrama de Componentes