Diseño:

Diagrama de alto nivel Interfaz:

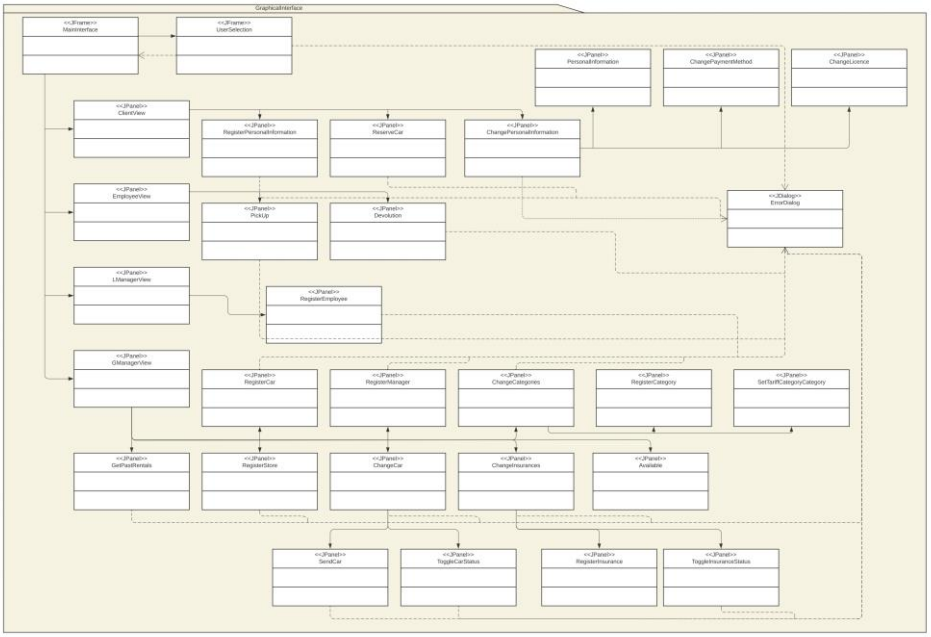
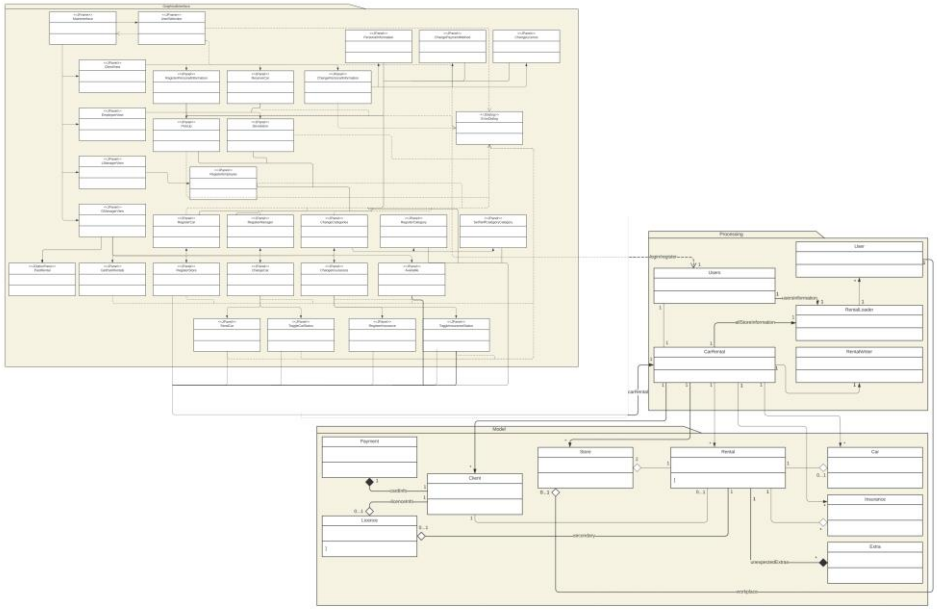


Diagrama de alto nivel:



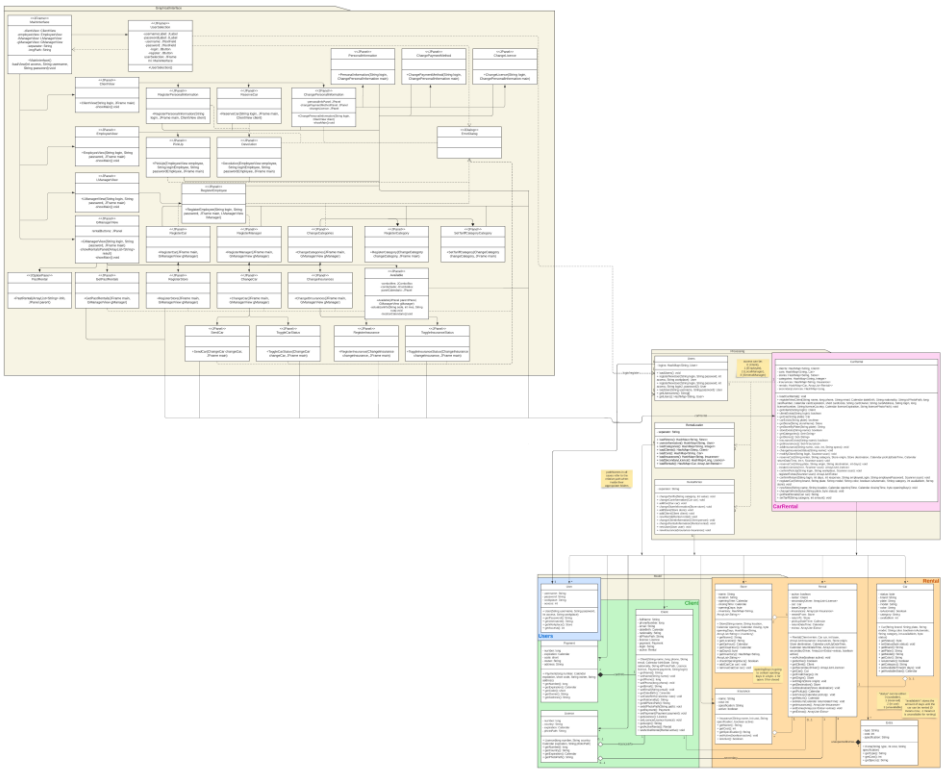
En primer lugar, la interfaz gráfica estará encargada de mostrar las distintas opciones según usuario. Se puede ver que para comenzar a usar la aplicación es necesario crear una conexión con Users para poder acceder a la información del usuario. Lugo se encargará de mostrar la información a la que su usario correspondiente tiene acceso. Por ello, la selección de usuario tiene una conexión directa con la interfaz, pues es quien le dice a la interfaz el tipo de usuario que ha iniciado sesión. Al seguir por la selección de usuario existe una nueva relación de realización con Users que tiene relación directa con RentalLoader, donde se guarda la información de tanto nuevos usuarios como existentes. Por

ello, RentalLoader tiene relación directa con User, pues dicha clase es la responsable de definir todos los objetos User con su respectiva información.

Cada Usuario se le mostrara su información con su panel asignado, ClientView, EmployeeView, etc. Y cada panel dentro de estos estará asociado a CarRental en el procesamiento. Esto porque CarRental contiene toda la información de la compañía y por ello es relevante que de ahí cada vista de usuario tenga definidos los atributos a los que puede tener acceso, pues el Local Manager solo debería poder ver los aspectos relacionados a su sede mientras el General Manager podría tener acceso a todo. Por esta misma razón CarRental tiene relación directa con RentalLoader y Users. Adicionalmente, también tiene relación con RentalWriter, pues mediante esta clase, los usuarios con la jerarquía necesaria pueden realizar ajustes sobre la compañía.

CarRental establece las conexiones con varias de las clases del modelo, dado que la compañía justamente maneja un sistema de rentas con distintas tiendas y vehículos, con clientes y seguros asociados en cada caso. Por ello, viendo la clase Rental, esta tiene relaciones de agregación con una tienda, un seguro, una licencia y un vehículo. Dicha relación de agregación existe dado que la renta de cualquier carro hecha en cualquier tienda finaliza, lo que quiere decir que, aunque esta desaparezca, los objetos Store, Insurance, Licence y Car seguirán existiendo. A su vez tiene relación de composición con los costos Extra, pues estos si dependen de una renta existente para seguir existiendo, dado que surgen a causa del propio alquiler. La clase cliente tiene una relación de agregación con la licencia, dado que esta no depende únicamente del cliente para poder existir, a causa de que puede haber licencias secundarias para otros conductores del vehículo rentado. Sin embargo, el cliente tiene una relación de composición con la clase Payment, pues la información de pago del cliente depende justamente de él para existir. Finalmente, regresando a la clase User en el procesamiento, esta tiene relación directa con Stores en el modelo, pues los usuarios pueden estar asignados como empleados a una tienda específica.

Diagrama de clases:



Responsabilidades:

Responsabilidad	Componente
Seleccionar el usuario para acceder a la aplicación	GraphicalInterface
Acceder a las opciones de cliente (ClientView): registrar información personal, hacer una reserva, cambiar información personal (Datos personales, método de pago, licencia)	
Acceder a las opciones de empleado (EmployeeView): Asignar recoger un vehiculo, asignar devolver un vehiculo	
Acceder a las opciones de Local Manager (LManagerView): registrar un empleado nuevo	

Acceder a las opciones de General Manager (GManagerView): registrar un vehículo, registrar tienda, registrar local manager, cambiar carro de sede y estado, registrar categoría y cambiar precio de categoría, registrar seguro y cambiar estado de seguro, ver historial de rentas pasadas y ver cantidad de vehiculos disponibles para una sede.	
Cargar usuario	Processing\Users
Crear nuevo usuario	
Cargar carRental	Processing\CarRental
Registrar nuevo cliente	
Verificar que un cliente existe por su login	
Registrar un vehículo	
Verificar si un carro existe según su placa	
Rserevar un carro	
Confirmar recoger un vehículo	
Confirmar devolver un vehículo	
Encontrar un cliente	
Encontrar una tienda por placa de carro dentro de su inventario	
Modificar un cliente	
Encontrar una tienda por su nombre	
Encontrar todas las tiendas	
Verifica si la tienda existe según su nombre	
Encontrar un vehículo por su placa	
Crear una nueva tienda	
Cambiar el estado de un vehículo dado su placa	
Definir la tarifa de un servicio	
Obtener todas las categorías existentes	
Obtener todos los seguros existentes	
Verificar si el seguro existe dado su nombre	
Añadir un nuevo seguro	
Cambiar el estado de un seguro dado su nombre	
Registrar extra una renta	
Conseguir historial de rentas	
Cargar archivos de: información de usuarios, clientes, vehículos, tiendas, rentas hechas y seguros.	Processing
Escribir o reescribir la información dentro de los archivos según cambios	
Al hacer una renta se asocia el cliente que la hace con su información pertinente: nombre, teléfono, correo, fecha de nacimiento, nacionalidad, licencia, log in y estado de la renta	Model\Client
Se asocia la renta activa con el cliente	
Al cliente se le agrega información completa de su licencia: número, país, fecha de vencimiento y foto	
La información del pago del cliente (datos de la tarjeta): número, fecha de vencimiento, código, propietario, dirección	

Se crea la renta de un vehículo: estado, cliente, conductor secundario en el caso de haberlo, vehículo, cargo base, seguro, tienda de donde se rentó, tienda donde se devuelve, fecha para recogerlo, fecha para devolverlo y cargos extras en el caso de haberlos.	Model\Rental
A la renta se le agrega la información completa del vehiculo: estado, marca, placa, modelo, color, transmisión, disponibilidad en cuantos días, disponibilidad en fecha.	
A la renta se le agrega la información completa del seguro: nombre, costo, especificaciones y estado.	
Para saber el vehiculo de donde se renta y donde se devuelve se le agrega la información completa de dicha tienda: nombre, ubicación, hora de apertura, hora de cierre, días en los que abre.	
Se definen los cargos extras que se llegan hacer: tipo, costo, especificación.	
Se crea un usuario con su unername, password, workplace (null en caso de ser client) y access(0:client, 1: employee, 2:localManager, 3:generalManager)	Model\User