**Diseño proyecto 2**

Cuentas predeterminadas para prueba:

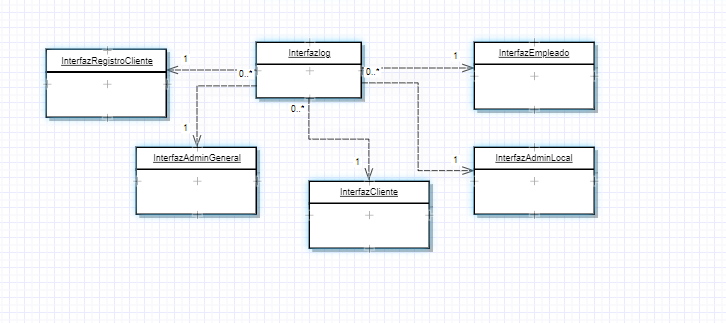
Usuario cliente: g.chaparr contraseña:12345

Usuario administrador general: generalprueba contraseña:12345

Usuario administrador local: localprueba contrasela

En el cuadro al lado de usuario ingresar el tipo de usuario: administrador general, administrador local y cliente

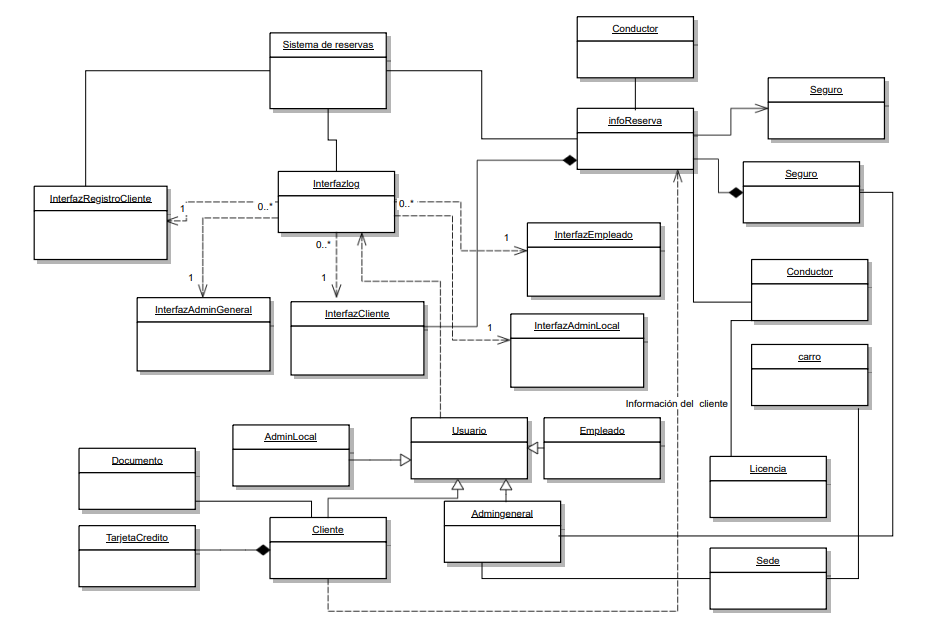
**1.Diagrama de las clases de la interfaz**



En este diagrama podemos ver como las clases se relacionan con la clase interfazLog , pero no se relacionan con ninguna otra clase. Esto se debe a que las demás clases dependen de la clase interfazLog para ser creadas, pero entre ellas no se relacionan ya que solo se relacionan con las clases de la lógica del sistema mediante las cuales hacen uso de métodos de las diferentes clases.

**2.Diagrama de alto nivel**

Ahora veremos un diagrama de alto nivel que enseña las relaciones de las clases de la interfaz con las clases de la lógica



**2.Toma de decisiones para la implementación del proyecto:**

**Usuario**: vimos que existían varias clases que requerían de unos métodos en común, por lo que creamos la clase Usuario que es abstracta. Tiene como atributo 2 strings, login y password. Tiene un método llamado entrar que recibe login, password y una lista de instancias respectivamente, en el método primero se define una variable booleana que por defecto es false, luego se obtiene la posición inicial de la lista y se compra con el login, en caso de que sea diferente, se vuelve a evaluar con la siguiente posición; cuando se comprueba que existe el login en la lista, se evalúa si es correcta la contraseña de la instancia con el ingresado, cambiando la variable a true y terminando el ciclo, finalmente se retorna la variable

**Admin\_General**: esta hereda de usuario, tiene 2 métodos

-nuevo\_car: recibe como parámetro las características del carro, además de la lista donde se guardan los carros, luego llama al constructor de la clase carro y le da los parámetros.

-carro\_fuera: recibe la placa del carro que desea borrar y la lista general, itera la lista para obtener la instancia del carro deseada y lo remueve de la lista

**Admin\_local:** esta hereda de usuario, tiene 2 métodos:

-nuevo\_empleado: recibe login, password y la lista general de empleados, llama a la constructora de empleado y finalmente lo añade

-cambio\_datos: recibe el login, password y la lista general de empleados, busca por posición en la lista y obtiene el login de la instancia, repite el mismo proceso en la siguiente posición si falla en la comparación del login de la instancia con el recibido, si es el mismo se cambia la contraseña con la función set password

-empleado hereda de usuario, tiene 2 métodos:

*estado\_carro*: recibe placa, estado y la lista general de carro, busca en la lista por la placa de carro y cambia su estado

**cliente**: hereda de usuario, además de los atributos heredados, también tiene como atributo nombre, correo, celular y una lista.Tiene como métodos los geters y setters de los atributos

**Decisiones en el diseño de interfaces:**

Para el diseño de interfaces, en un principio pensamos en hacer para cada uno de los requerimientos funcionales una ventana, en la cual se pudieran ejecutar cada uno de ellos, pero finalmente decidimos agrupar los requerimientos funcionales por tipo de usuario, como cliente, administrador general, administrador local y empleado. Cada usuario tiene su respectiva interfaz donde podrá hacer las acciones que le pertenecen en el contexto del sistema de reservas. La interfaz que le pertenece a cada usuario será delegada por la InterfazLog en la cual se hará la validación del registro del usuario en el sistema. Es por esto que finalmente decidimos implementar 5 interfaces:

InterfazLog:

Esta interfaz se encargará de validar el tipo de usuario y si está registrado en el sistema. De no ser así el usuario podrá registrarse en esta ventana.

InterfazCliente:

En la interfaz cliente se podrán hacer reservas y modificar la información de reservas. Si el cliente tiene reservas, estas se enseñarán en un en un componente JList.

InterfazAdminGeneral:

En esta interfaz el administrador podrá registrar seguros y carros nuevos.

InterfazAdminLocal:

En esta interfaz el administrador local podrá hacer cambios en la información de los carros y de los empleados de las diferentes sedes.

InterfazEmpleado:

En la interfaz de empleado se podrá informar del estado de un carro, el registro de nuevos conductores, y la entrega y devolución de carros.

