GIVAM: Sistema de Gestión Integente de Vehículos y Administración de Mantenimientos

Actividad Integradora:

***Alumno***: Mariscal Rodríguez Omar Jesús

***Código de Estudiante:*** 220858478

***Maestra:*** Sabrina Lizeth Vega

***Fecha de Elaboración:*** 11 de Mayo de 2025

***Programación orientada a Objetos***

***Clave: I5289 Grupo: 195114***

***Martes y Jueves: 11:00am a 12:55pm***

Contenido

[Delimitación del Problema 3](#_Toc197729358)

[Título del Problema 3](#_Toc197729359)

[Delimitación del Problema 3](#_Toc197729360)

[Planteamiento del Problema Y Nombre del Programa 3](#_Toc197729361)

[Planificación con el Diseño del Programa 5](#_Toc197729362)

[Diagrama de Clases 7](#_Toc197729363)

[Diccionario de Clases 8](#_Toc197729364)

[Diagrama de Casos de Uso 11](#_Toc197729365)

[Diagrama de Comunicación 12](#_Toc197729366)

[Diagrama de Actividades 13](#_Toc197729367)

[Implementación de la Solución 19](#_Toc197729368)

[Login y Registro de GIVAM 19](#_Toc197729369)

[Gestión de Vehículos 22](#_Toc197729370)

[Registrar un Vehículo 22](#_Toc197729371)

[Editar un Vehículo 24](#_Toc197729372)

[Buscar un Vehículo 25](#_Toc197729373)

[Eliminar un Vehículo 25](#_Toc197729374)

[Mostrar el Registro de Todos los Vehículos 26](#_Toc197729375)

[Gestión de Usuarios 26](#_Toc197729376)

[Buscar un Usuario Registrado 26](#_Toc197729377)

[Mostrar Registro de Usuarios Registrados 27](#_Toc197729378)

[Gestor de Mantenimientos 27](#_Toc197729379)

[Registrar un Nuevo Mantenimiento 27](#_Toc197729380)

[Buscar Mantenimiento por ID 29](#_Toc197729381)

[Consultar Historial por Vehículo 29](#_Toc197729382)

[Editar un Mantenimiento 30](#_Toc197729383)

[Reportes 31](#_Toc197729384)

[Exportar un Reporte Determinado 31](#_Toc197729385)

[Exportar Reportes de un Vehículo 31](#_Toc197729386)

[Exportar Mantenimientos Entre Fechas 32](#_Toc197729387)

[Configuración de su Cuenta 33](#_Toc197729388)

[Información de GIVAM 34](#_Toc197729389)

[Cerrar Sesión 34](#_Toc197729390)

[Control de Almacenamiento 34](#_Toc197729391)

[Conclusiones de la Actividad 36](#_Toc197729392)

# Delimitación del Problema

Para este Producto Integrador, se pondrá en práctica ***todos los conocimientos*** y el viaje a lo largo de este curso, pasando por los ***pilares de la Programación Orientada a Objetos*** y todas las ***herramientas que hemos analizado*** para darle ***solución a una problemática determinada***; a continuación, se delimita la problemática que se resolverá y la implementación de la misma.

## Título del Problema

"Falta de un ***sistema integral*** para la ***gestión segura y eficiente*** de ***usuarios, vehículos y registros de mantenimiento*** automotriz"

## Delimitación del Problema

Actualmente, muchos talleres automotrices, flotas vehiculares y pequeñas empresas de mantenimiento ***carecen de herramientas digitales que les permitan gestionar de manera segura, eficiente y estructurada la información crítica*** relacionada con sus operaciones. En particular, los sistemas tradicionales (o la ausencia de ellos) presentan dificultades para:

* ***Registrar usuarios y distinguir sus roles*** (por ejemplo, administradores, técnicos o usuarios generales).
* ***Controlar los permisos de acceso y edición de información sensible***, como datos de vehículos o historial de mantenimientos.
* ***Mantener la trazabilidad de los servicios realizados*** a los vehículos.
* Organizar y consultar el historial de mantenimientos de manera rápida y segura.
* ***Generar reportes de manera rápida*** sobre los distintos ***mantenimientos que se le han realizado*** a sus vehículos.

Estas limitaciones conllevan una ***falta de control, errores humanos frecuentes, duplicidad o pérdida de información*** y, en general, ***ineficiencias*** que afectan directamente la calidad del servicio, la administración de los recursos y la satisfacción de los clientes.

## Planteamiento del Problema Y Nombre del Programa

La ausencia de un ***sistema de gestión especializado*** con control de usuarios, y en consecuencia, la realización de estas tareas de forma manual provoca una serie de problemáticas específicas:

* ***No se puede restringir adecuadamente quién tiene acceso a información crítica***, permitiendo que cualquier usuario pueda ***modificar, eliminar o introducir datos importantes*** sin supervisión.
* El ***registro y seguimiento de mantenimientos*** de los vehículos resulta ***disperso***, ***inconsistente o vulnerable a manipulaciones***, dificultando la construcción de un historial confiable de servicios.
* El proceso de ***generación de reportes sobre mantenimientos***, vehículos o usuarios es ***manual, lento e ineficiente***, complicando la toma de decisiones estratégicas.
* La ***falta de estandarización en la entrada y almacenamiento*** de datos ***incrementa los errores humanos y disminuye la capacidad de escalar el servicio*** de forma ordenada.

Ante este contexto, surge la necesidad de utilizar la tecnología como una herramienta para automatizar y facilitar todos estos procesos manuales, ante los cuales, se ha desarrollado el siguiente programa de nombre:

***GIVAM***: Sistema de Gestión Inteligente de Vehículos y Administración de Mantenimiento

Este sistema se concibe como una solución para los conflictos antes mencionados y proporcionar una base sólida para una escalabilidad del negocio

* ***Proporcionar una plataforma digital estructurada*** para el registro, consulta y edición controlada de ***usuarios, vehículos y mantenimientos.***
* ***Distinguir niveles de acceso entre administradores y mecánicos*** mediante autenticación y validación de roles.
* ***Registrar y encriptar las contraseñas*** con las que se accedan al programa.
* ***Llevar el registro de los datos en formato .csv*** para facilitar su gestión y ***complementar la herramienta con programas como Excel*** para la visualización del almacenamiento del programa.
* Facilitar la ***generación de reportes y la exportación de información en formato .txt***, promoviendo la trazabilidad y el análisis de datos.
* Brindar un ***menú amigable*** que favorezca su uso incluso para ***personal con conocimientos básicos en informática.***

En resumen, ***GIVAM busca resolver la falta de control, seguridad y eficiencia en la administración de datos relacionados con usuarios, vehículos y mantenimientos dentro del contexto automotriz,*** optimizando así los procesos de gestión y mejorando la calidad operativa de los negocios que lo implementen.

# Planificación con el Diseño del Programa

El primer paso para la solución tecnológica es identificar el problema y delimitarlo para qué es puntualmente lo que queremos solucionar y/o automatizar, esta información está anteriormente en este documento, ahora, el desarrollo de GIVAM se dividió en 4 fases principales; las primeras fases del desarrollo corresponden al núcleo del programa, lo que esencialmente necesita para su primer puesta en marcha, y las demás son las funcionalidades extra de interés que aportan nutritivamente a los focos que GIVAM pretende solucionar:

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Nombre de la Fase | Resumen | Puntos a Desarrollar |
| 1 | ***Proceso de Abstracción*** | Capturar los objetos principales del problema y cómo interaccionarán entre ellos | ***Abstracción Primaria***   * Delimitar qué objetos intervienen en cada parte de la solución * Pensar en qué métodos y cómo se comunicarán entre los objetos * Realizar una primera versión del diagrama de clases enfocada en la estructura principal |
| 2 | ***Desarrollo de la Estructura Central de Funcionamiento de GIVAM*** | Estas funcionalidades forman el núcleo del sistema y deben estar presentes sí o sí para que funcione correctamente: | ***Gestión de Usuarios***   * Registro de Usuarios con Roles (Administrados, Mecánico) * Login con Validación de Credenciales * Diferenciación de permisos por tipo de usuario   ***Gestión de Vehículos***   * Alta de Vehículos * Consulta General e Individual * Modificación de Vehículos * Baja de Vehículos   ***Gestión de Mantenimientos***   * Registrar Mantenimiento (preventivo y correctivo) * Asignar mantenimiento a un vehículo * Consultar mantenimiento por Vehículo   ***Persistencia de Datos***   * Guardar y cargar datos de vehículos, usuarios y mantenimientos en archivos .csv o .txt   ***Menú Interactivo***   * Menús dinámicos según el tipo de usuario * Validaciones básicas de entrada * Flujo lógico entre las funcionalidades del sistema |
| 3 | ***Lógica y edición refinada*** | Estas funcionalidades mejoran la experiencia y aportan más potencia al sistema, pero pueden desarrollarse después del núcleo | ***Historial y Seguimiento***   * Historial completo de mantenimiento por vehículo * Consulta de mantenimientos por fecha * Consulta de mantenimientos por tipo   ***Sistema de Búsqueda y Edición***   * Búsqueda vehículos por placa, tipo, marca * Buscar mantenimientos por ID o por los realizados a un Vehículo * Implementar la edición del propio perfil * Administradores sean capaces de editar datos de Vehículos y Mantenimientos Registrados |
| 4 | ***Seguridad y Reportes*** | Estas funcionalidades dan un plus profesional y podrían desarrollarse al final o como mejora posterior si el tiempo lo permite: | ***Seguridad***   * Encriptación de contraseñas mediante un Hash simple   ***Reportes y Exportación***   * Generar reporte de mantenimientos de un periodo específico * Reporte por vehículo específico * Exportar historial de mantenimiento a .txt |

Tener una planificación por fases mejora enormemente la organización de un proyecto grande, además de esto, el organizarlo de tal manera, que las funcionalidades primarias sean las primeras en desarrollarse, nos da un seguro contra el tiempo, esto es, si el proyecto tuviera que ser entregado por distintas circunstancias antes de tiempo sin desarrollar aún todas las funcionalidades que estaban previstas, el proyecto después de un punto crítico en el desarrollo (en el caso de GIVAM es la segunda fase), estaría funcionando, por lo que podría ser entregado aún si el tiempo no dio para planificar .

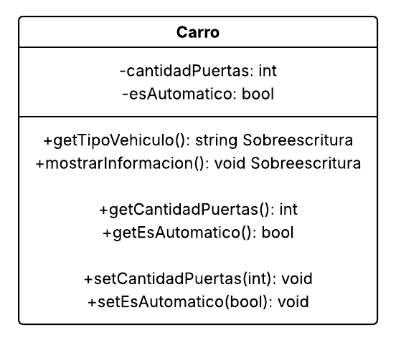
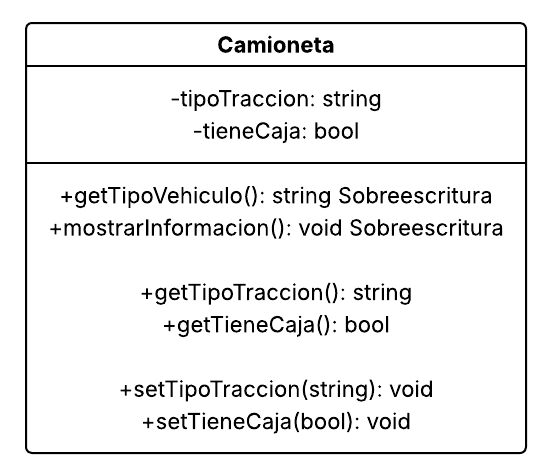
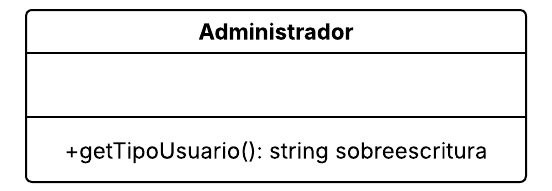
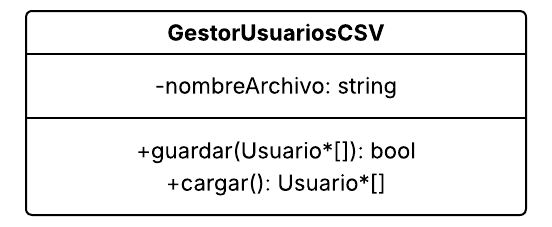
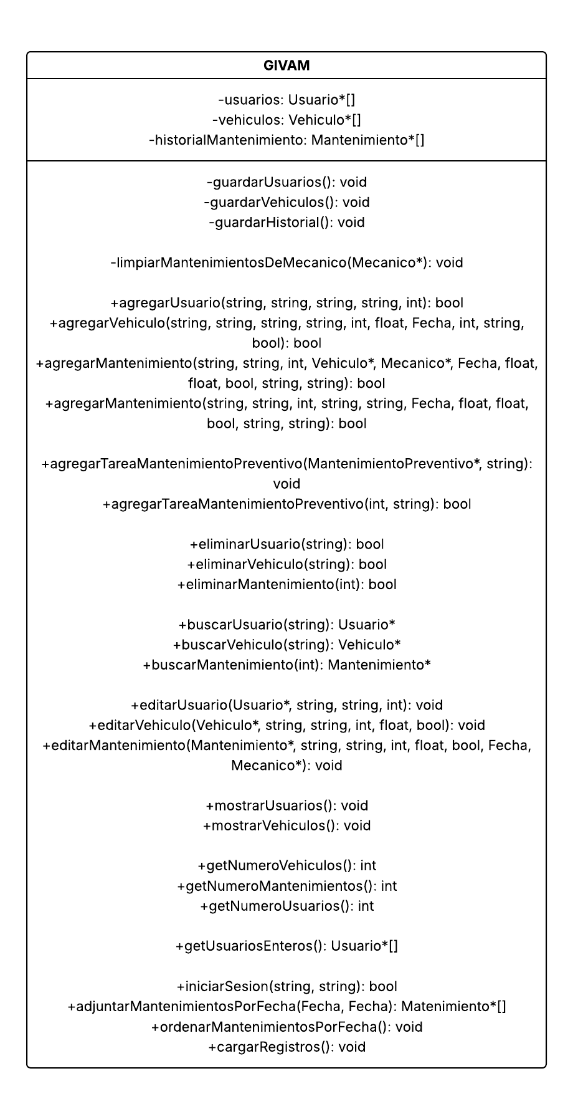
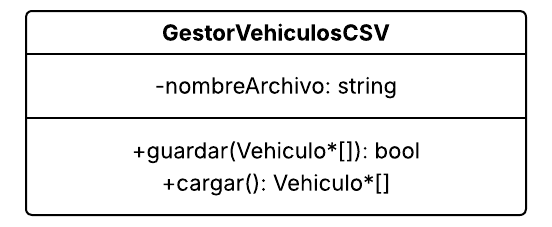
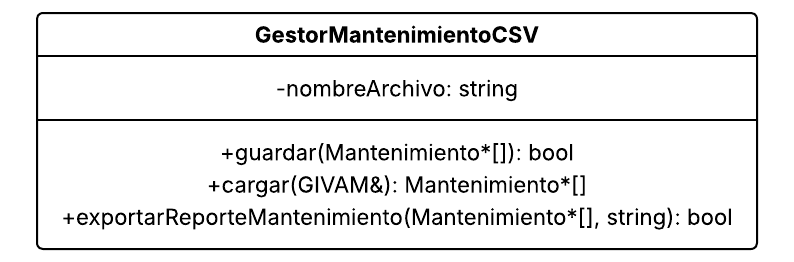
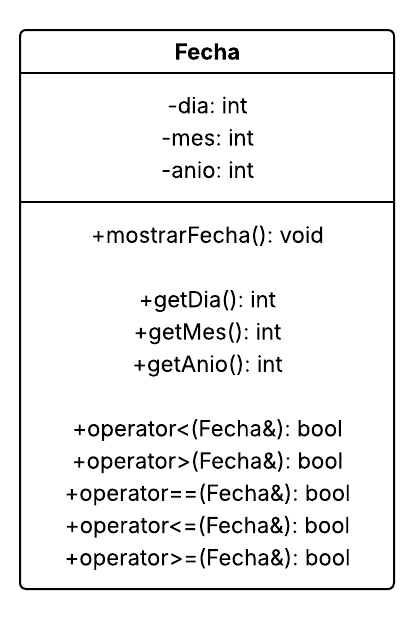
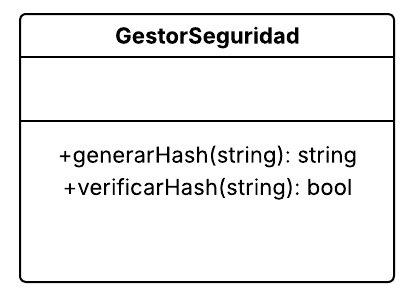
A continuación, se anexa un link al repositorio de GitHub donde, a partir de la primera versión del producto integrador, se fueron anexando los cambios registrados, llevando un control de las versiones, no solo del programa, sino de la documentación en general, incluyendo los diagramas y viendo con comentarios los cambios que ha sufrido el proyecto:

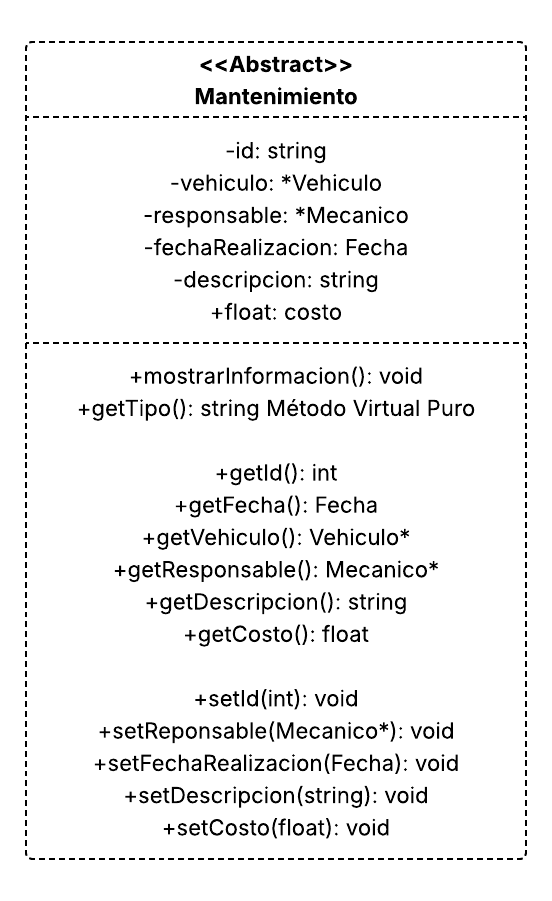
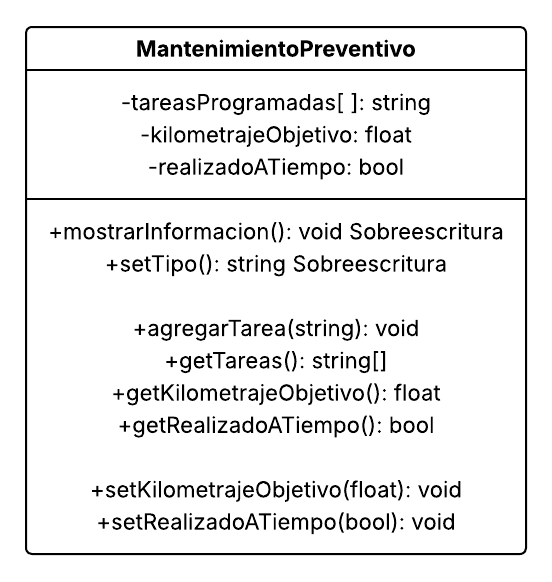
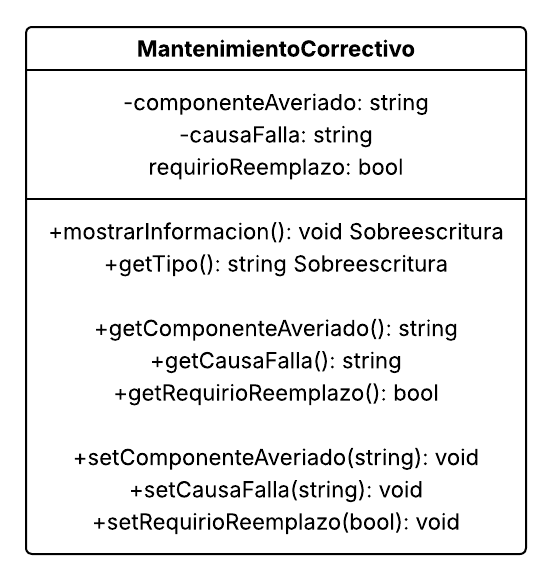
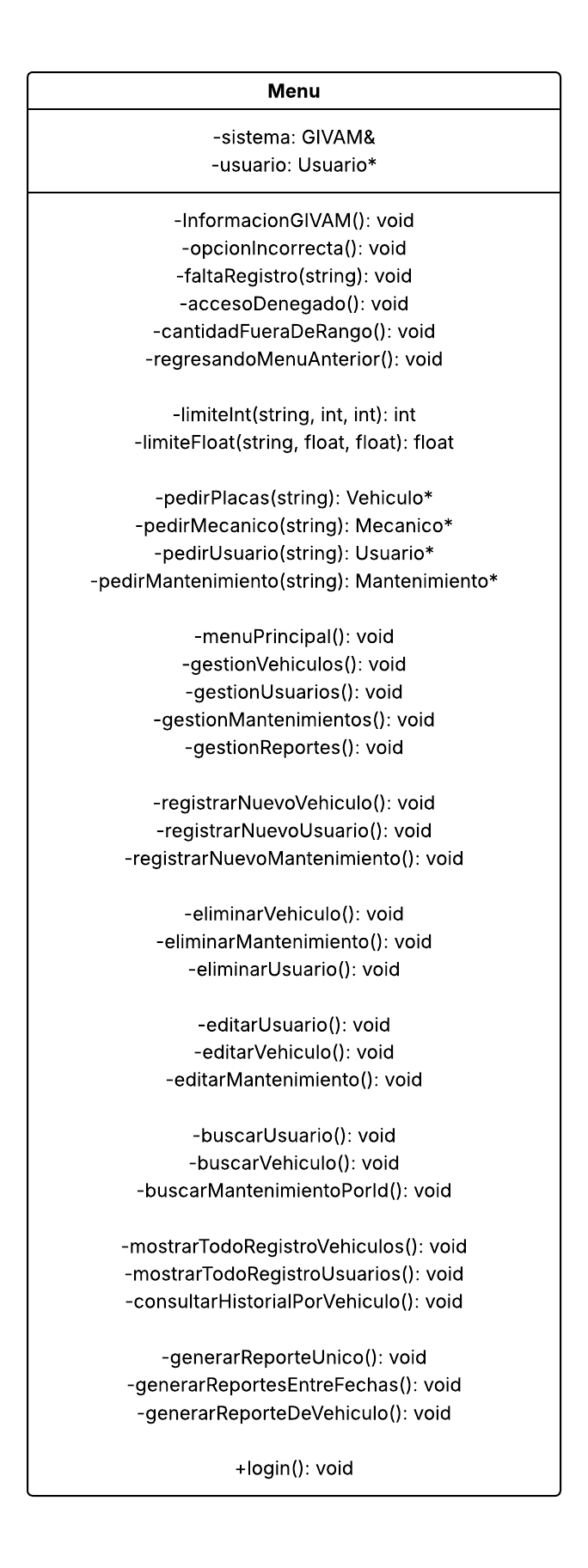
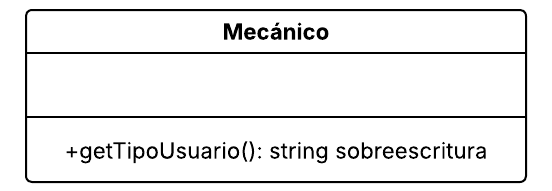
<https://github.com/OmarMariscal/Programaci-n-Orientada-a-Objetos>

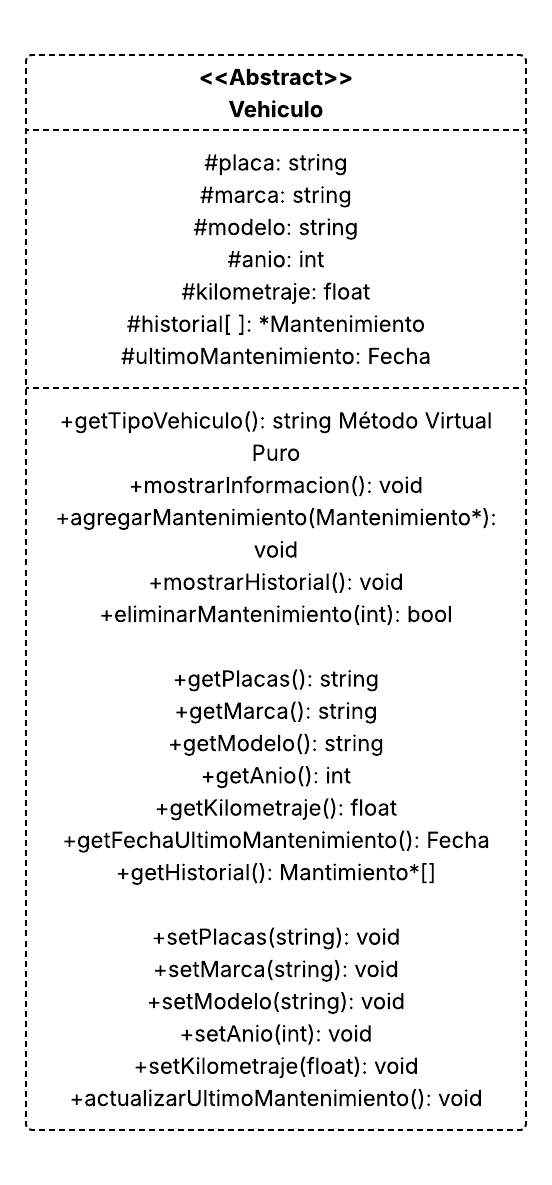
En suma a todo este proceso de planificación del desarrollo, entran los diagramas UML, una convención de modelado que nos ayuda mucho a clarificar y previsualizar cómo será el programa en distintos niveles y a través de distintas vistas, ayudando tanto a los clientes que verán que se supone que el programa cubre y que no, y a los desarrolladores para guiar su proyecto en el desarrollo, trabajar de manera colaborativa teniendo todos ya una idea más cerca de lo que globalmente el programa será y para que el funcionamiento del mismo sea más claro a otros programadores al explicar funciones específicas a nivel de comunicación, interacción etc, para su mantenimiento y expansión.

## Diagrama de Clases

## Diccionario de Clases

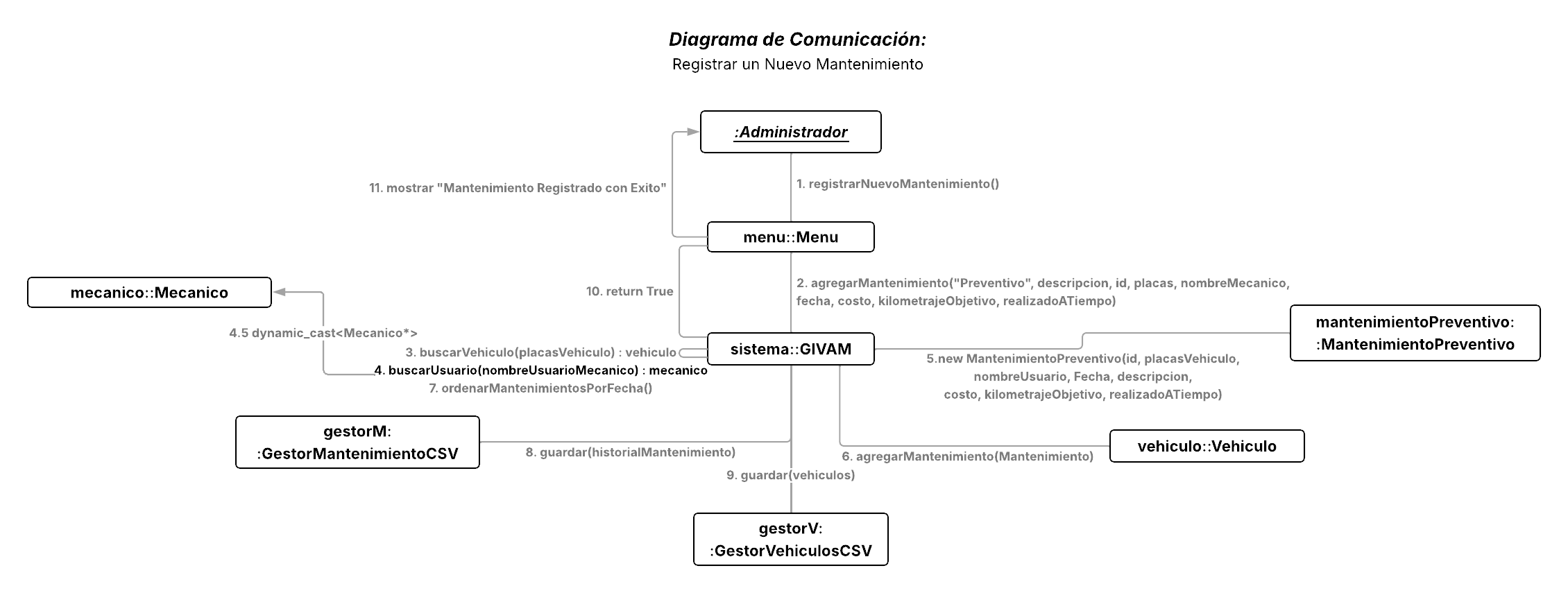






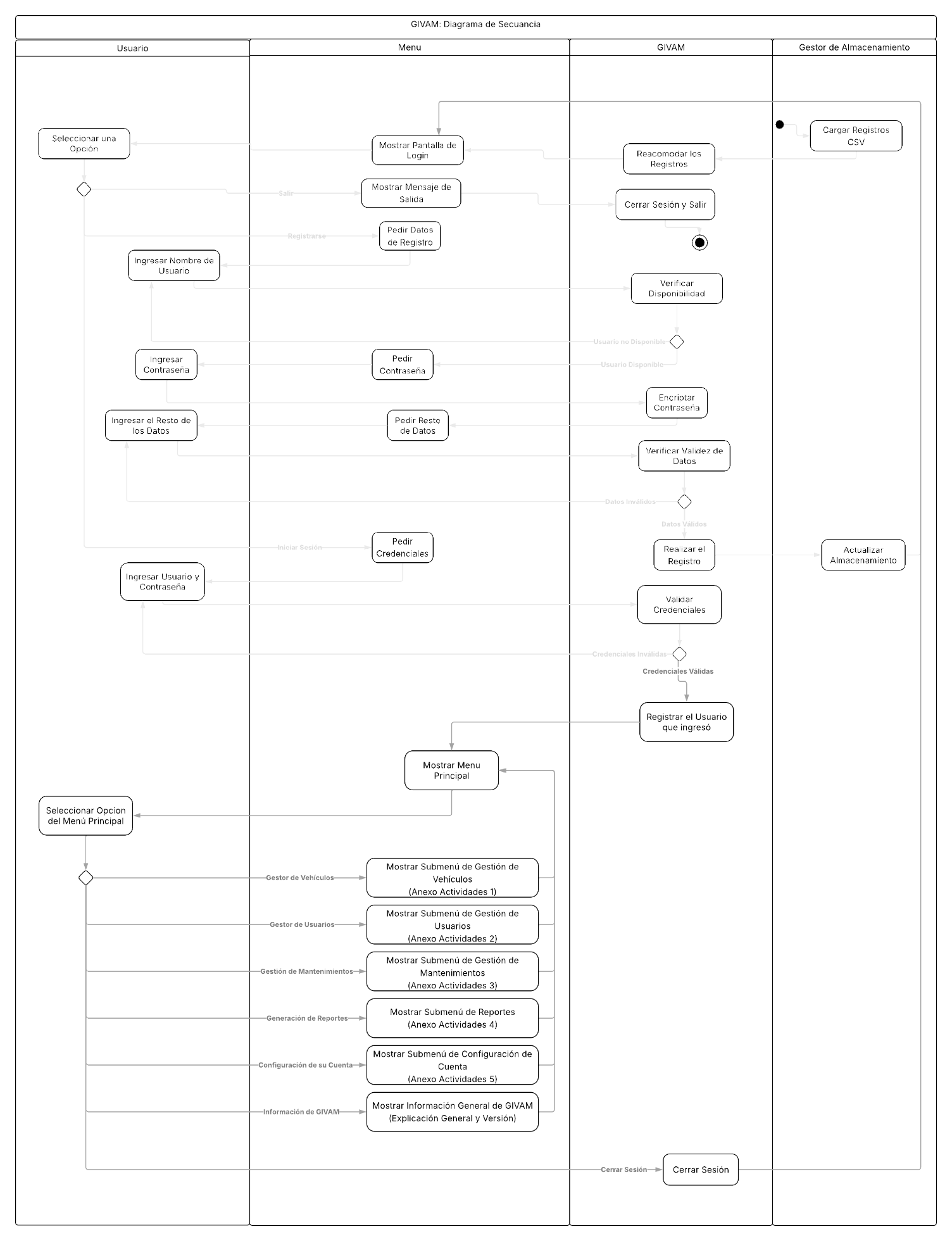
## Diagrama de Casos de Uso

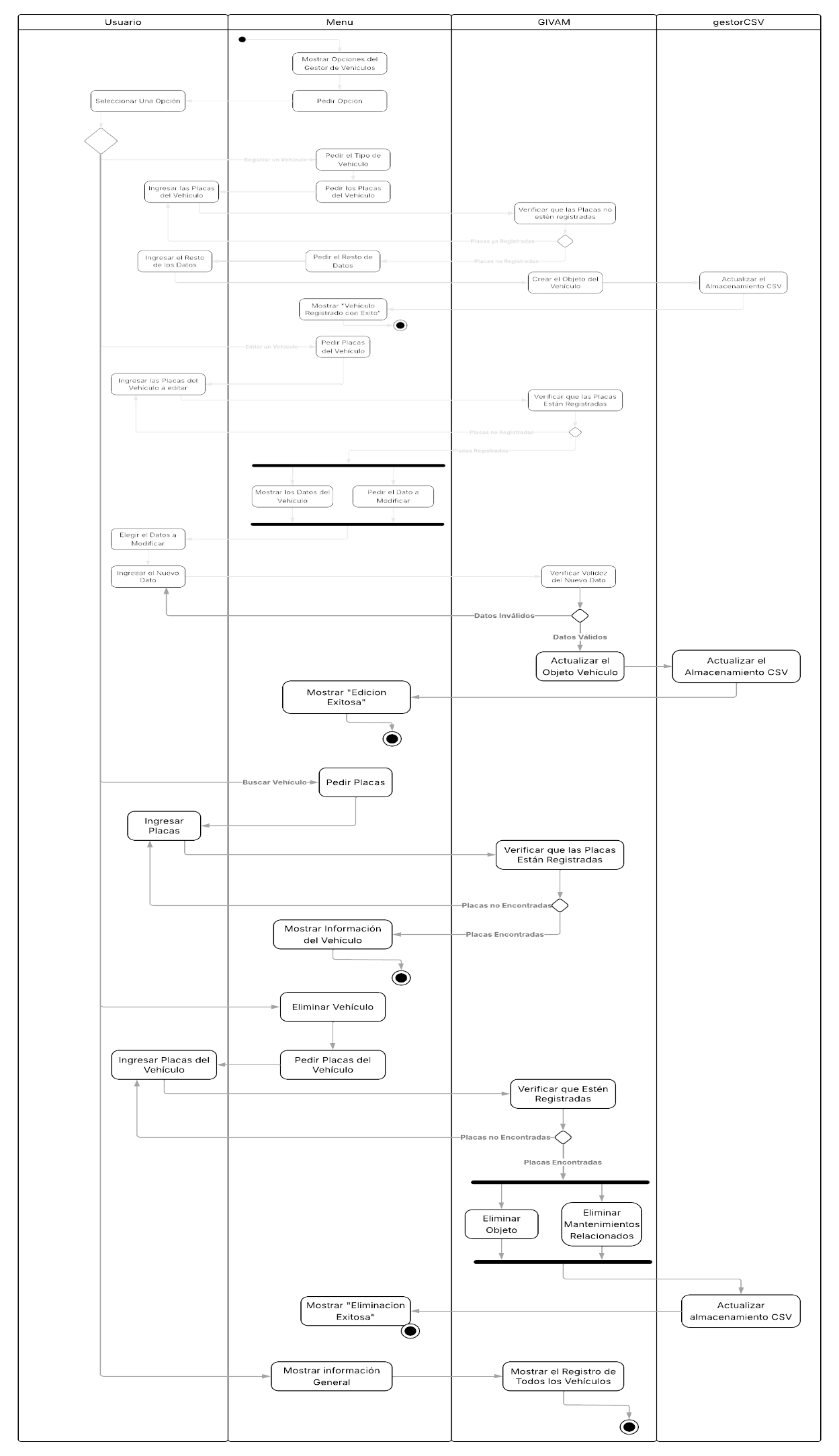
## Diagrama de Comunicación



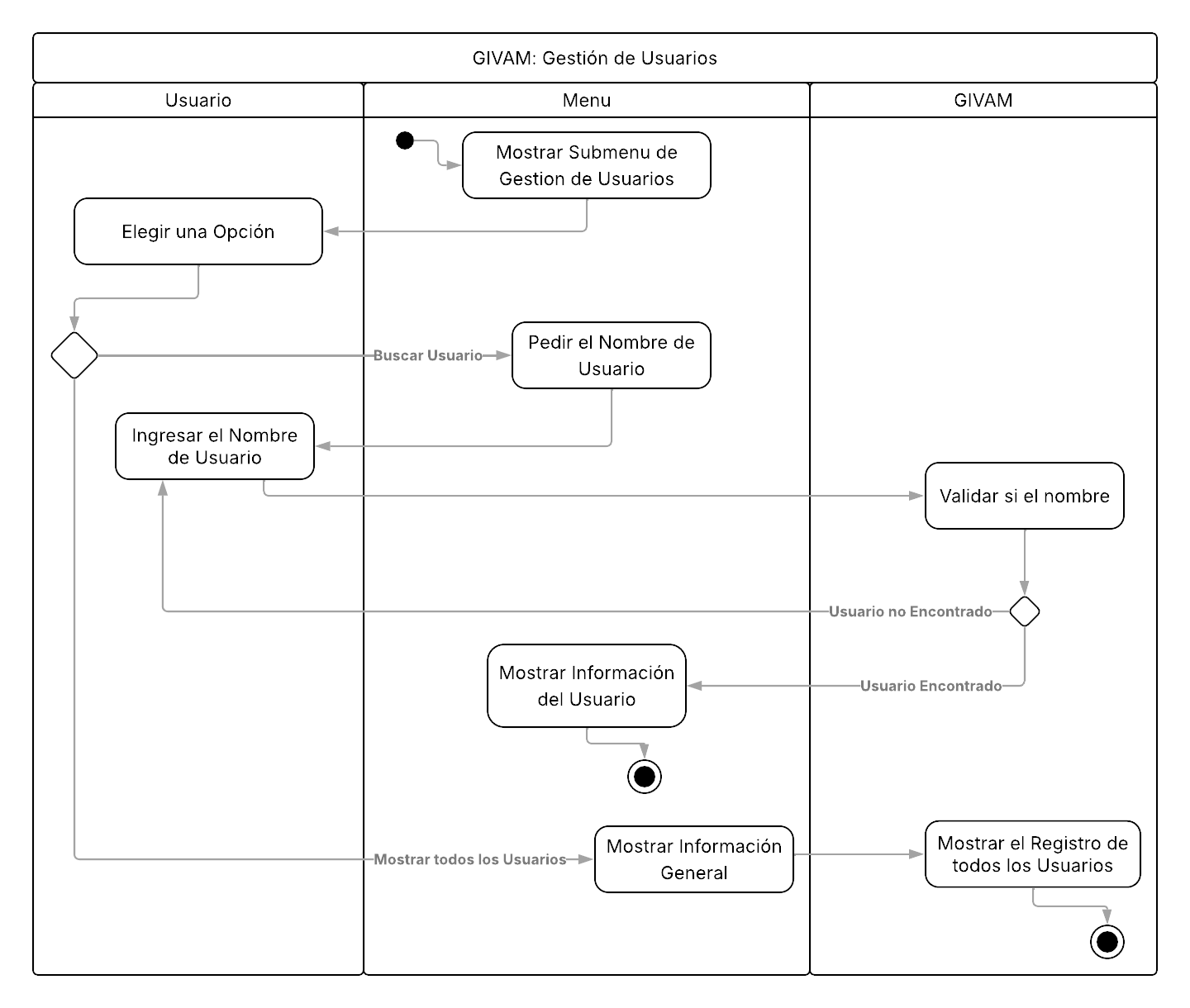
El diagrama organiza como es que los distintos objetos se comunican entre sí y qué mensajes se mandan para llevar a cabo la acción de registrar un nuevo mantenimiento e intervienen varias veces para que esta acción se lleve correctamente a cabo.

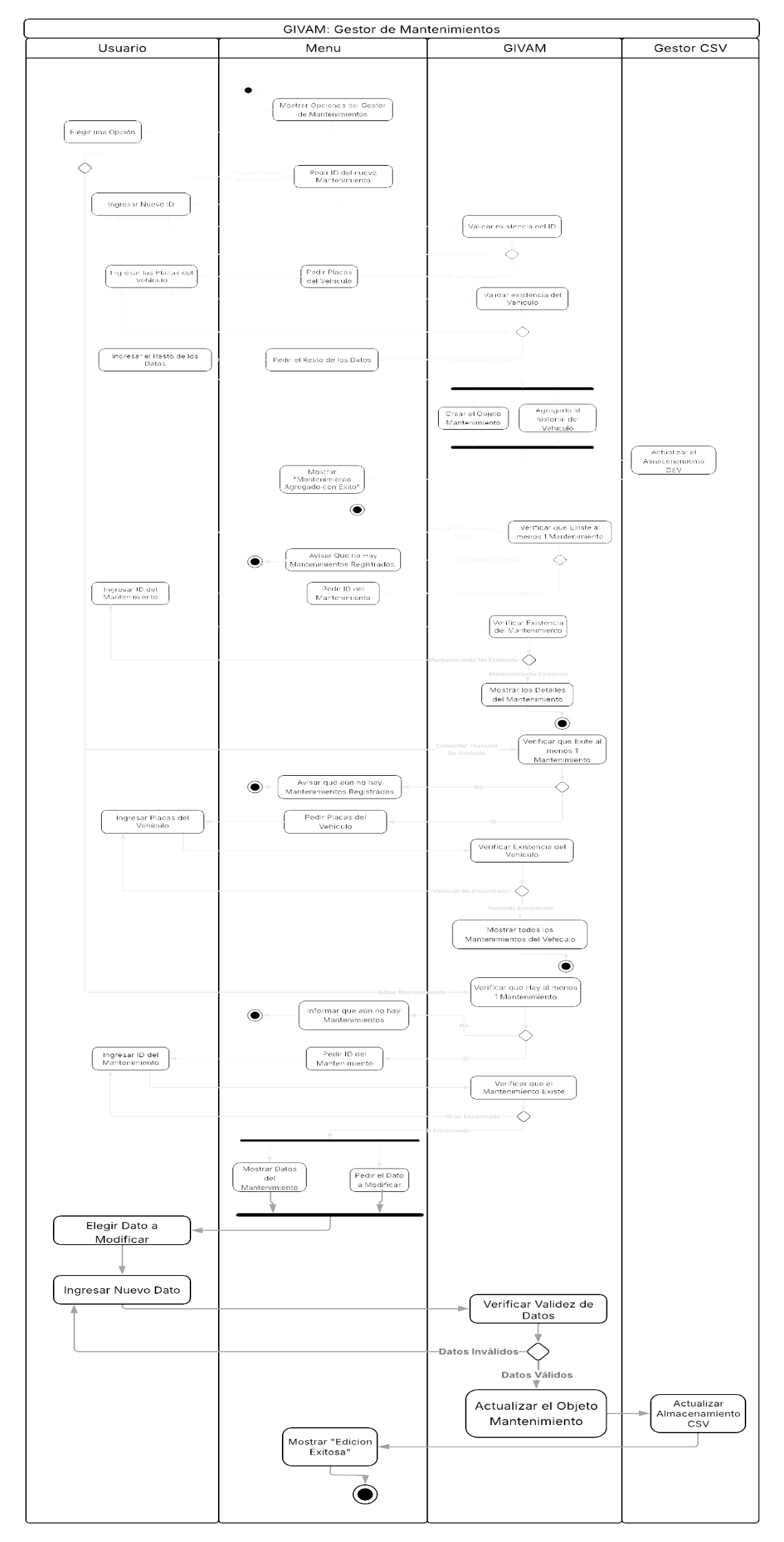
## Diagrama de Actividades

El diagrama de Actividades que a continuación se presenta, representa el flujo sintetizado del funcionamiento de todo el programa de GIVAM, por la extensión del mismo, el diagrama de actividades se ha segmentado en el diagrama principal que abarca desde la validación de credenciales, la elección del submenú hasta la salida del programa, los demás diagramas complementan las acciones de los distintos submenús:

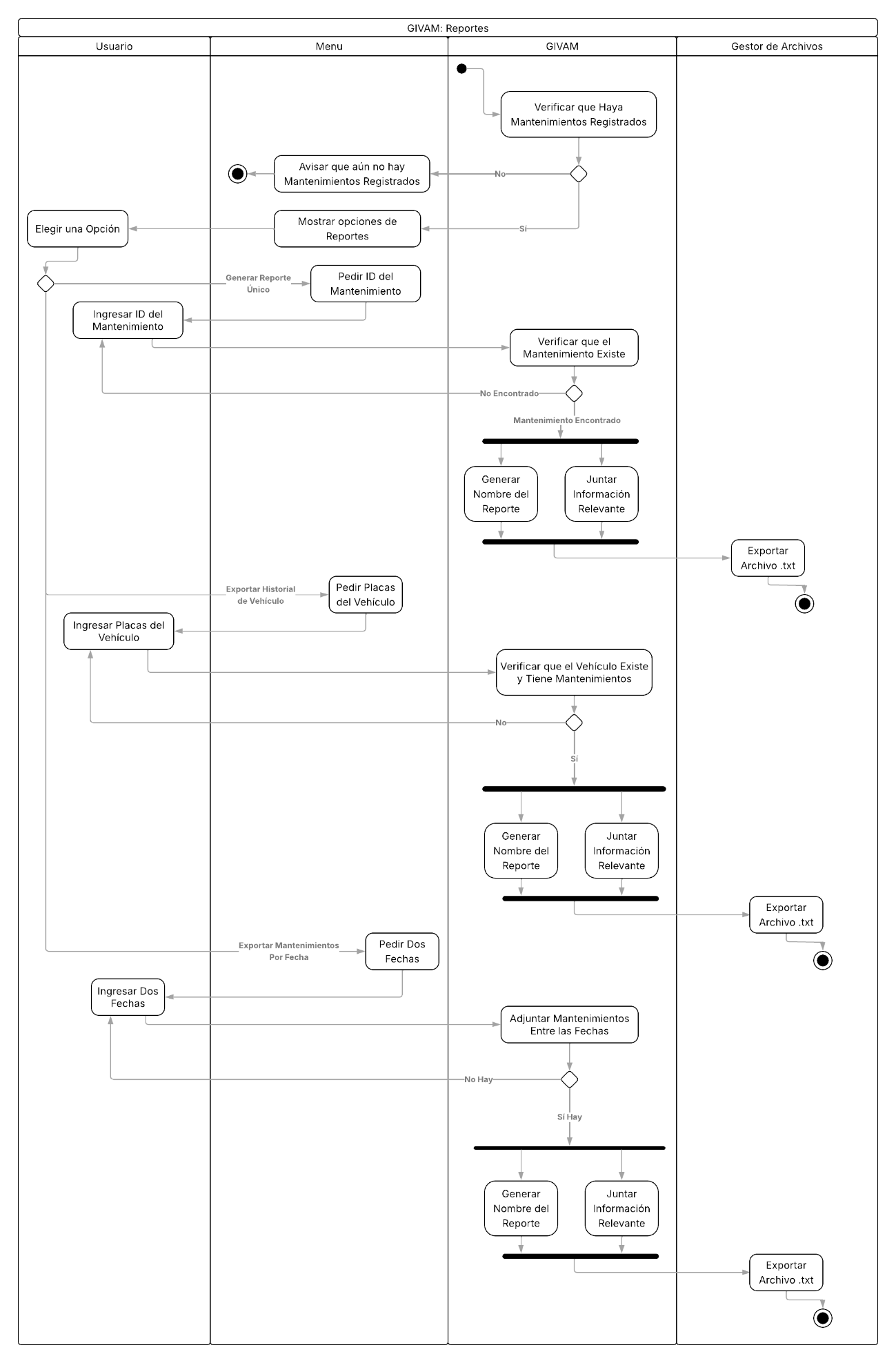
***Anexo Actividades 1:*** “Gestor de Vehículos”

Anexo Actividades 2: “Gestor de Usuarios”

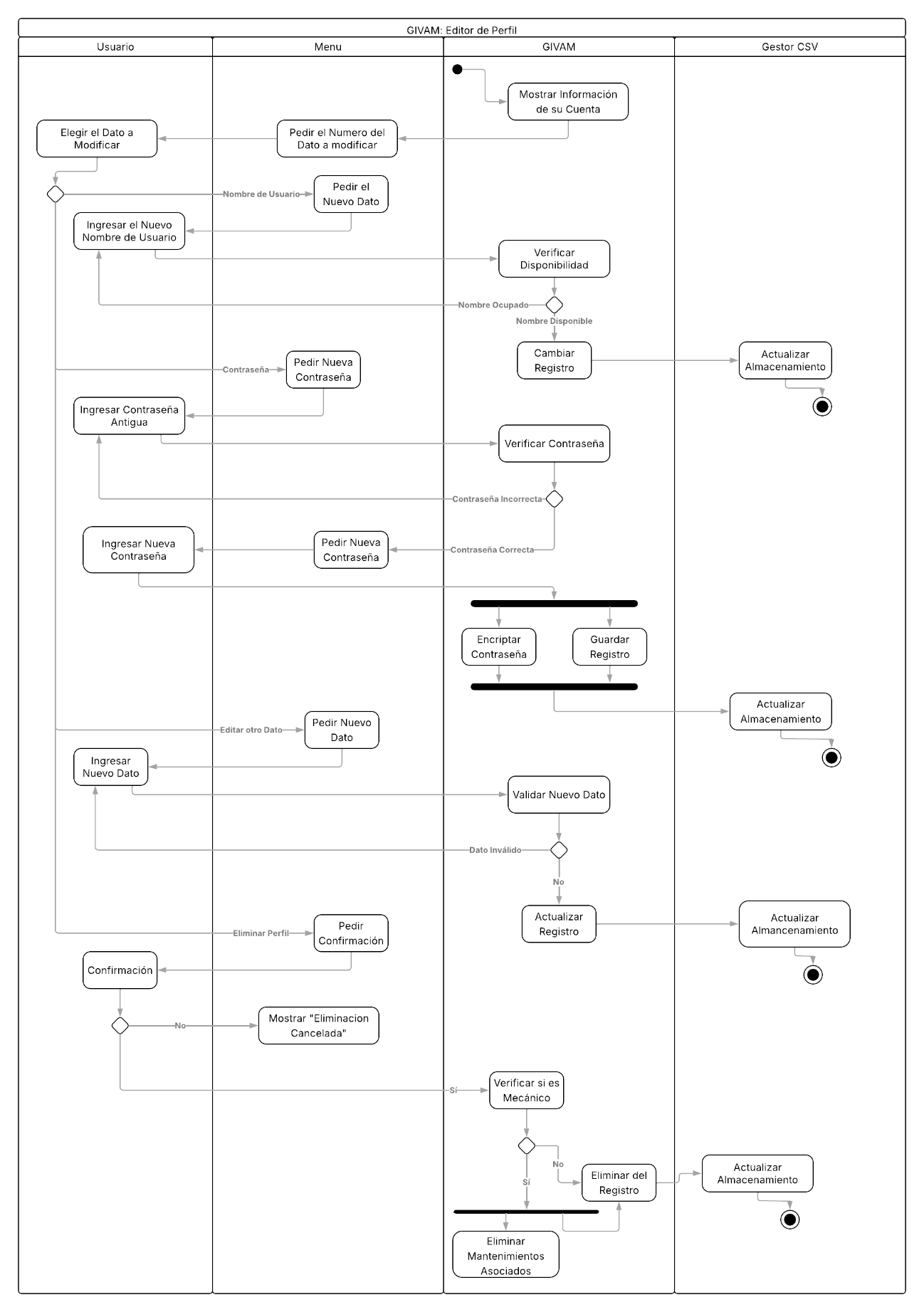


***Anexo Actividades 3:*** “Gestor de Mantenimientos”

Anexo Actividades 4: “Actividades de Reportes”



Anexo Actividades 5: “Configuración de la Cuenta”

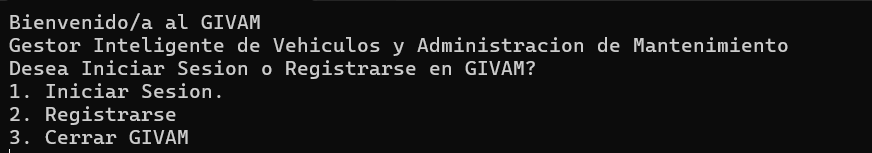


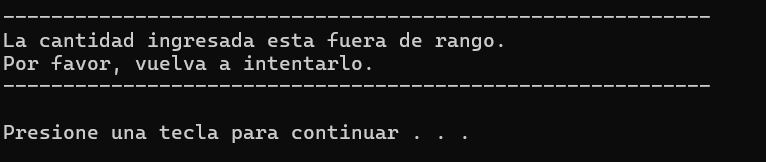
# Implementación de la Solución

Para este programa y aplicar los principios y conocimientos recabados de POO para GIVAM se ha utilizado el lenguaje C++, y a continuación se muestran capturas del programa corriendo:

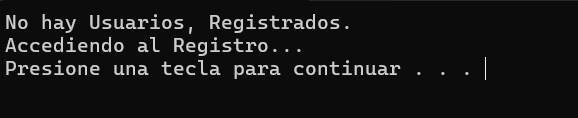
## Login y Registro de GIVAM

Iniciando el programa, GIVAM pone esta como primer menú:

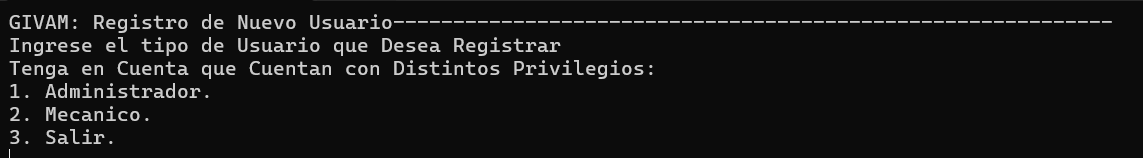
Dándote la bienvenida y presentándose distintas opciones ante usted; cabe destacar que el programa viene preparado para si en cualquier menú pone una opción inválida (alguna que no se presente como por ejemplo podría ser 4 en este caso), se muestra el siguiente mensaje de error:

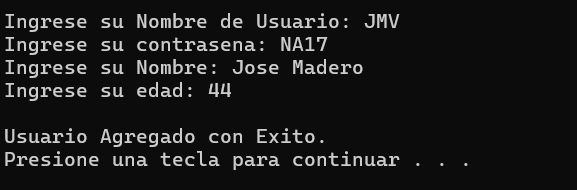


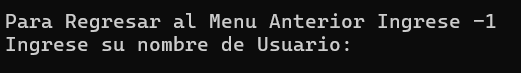
Y se regresa al menú anterior, así, evitando errores o casos inesperados de números fuera de rango.

Al ser el primer puesta en marcha del programa y no tener registros aún, si intentamos iniciar sesión aparecerá lo siguiente:

Como no hay usuarios para acceder, en automático nos pasa al registro de perfil, que es el mismo menú si hubiéramos presionado la segunda opción para registrarse.

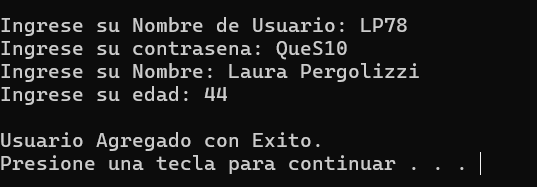
Aquí podemos seleccionar el tipo de usuario con el que nos registraremos, tenga en cuenta que los administradores son los que tienen un control total del programa como para editar registros por ejemplo, mientras que un mecánico no tiene tales privilegios; sin embargo y bien como se muestra en el diagrama de clases, GIVAM se compone de Usuarios, así que para que el programa funcione bien, además de administradores necesitamos mecánicos, ya que a ellos se les hará responsables de los mantenimientos registrados, registremos primero un administrador con la opción 1:

Al ingresar los datos que piden (con restricciones en la edad para un mínimo, por ejemplo), nos avisa que el usuario ha sido agregado con éxito y con ello, nos mandará al apartado de Login donde podremos acceder al sistema:

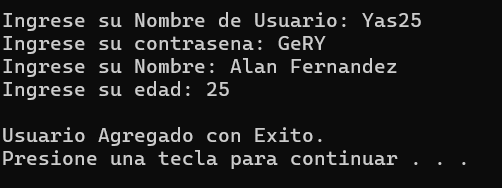


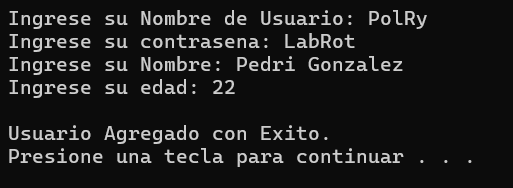
Sin embargo, antes de entrar a GIVAM, registraré un par de usuarios más para demostrar bien su funcionamiento:

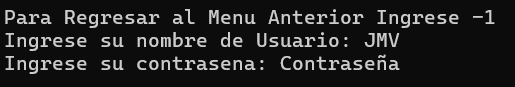
Primero a la Administradora Laura:

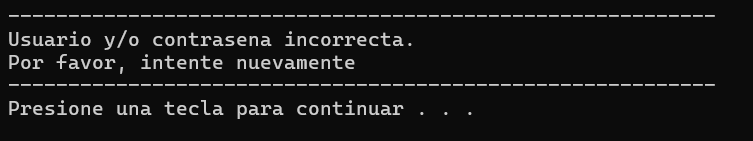


Y a los mecánicos Alan Fernandez y Pedri Gonzalez:

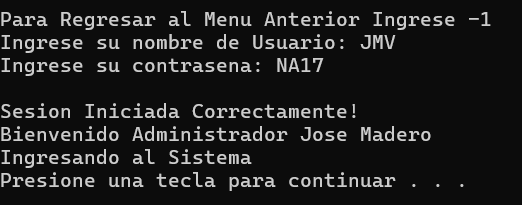


Y ahora sí, paremos al apartado de Login, se nos preguntarán las credenciales, que son el nombre de usuario y la contraseña, y en caso de ser ingresadas erróneamente se mostrará el siguiente mensaje:

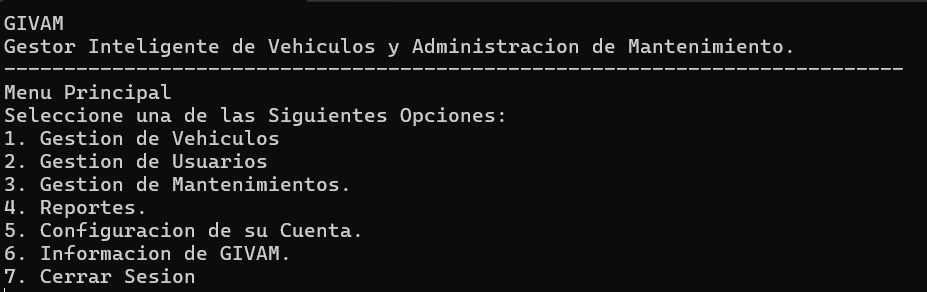




Pidiéndonos nuevamente las credenciales, si las ingresamos correctamente, se nos da un mensaje de bienvenida con el nombre registrado.

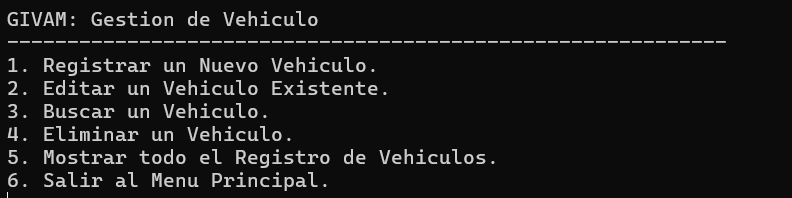


Entrando así a GIVAM y mostrándosenos el Menú Principal:

Desde aquí tenemos el control al sistema, dentro de algunos submenús, habrá opciones de las cuales, solo los usuarios del tipo administrados pueden disponer; veremos aquello cuando toque alguna de esas opciones.

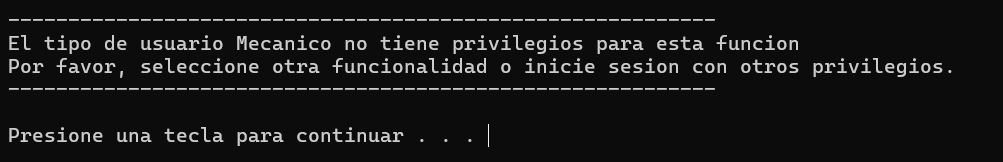
## Gestión de Vehículos

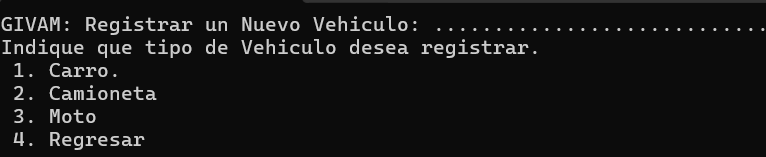
Seleccionando el primer menú tenemos el despliegue del submenú: Gestión de Vehículos:

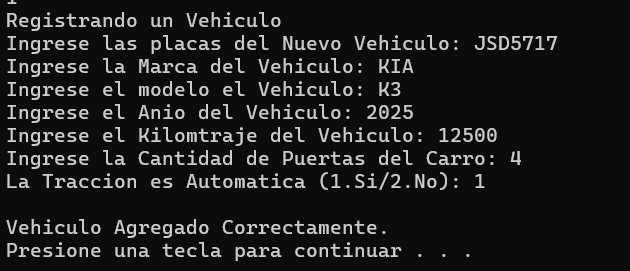


### Registrar un Vehículo

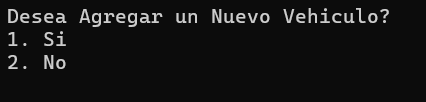
Intentado acceder a la primera opción, para registrar un nuevo vehículo, es una funcionalidad reservada para los administradores, si intentamos acceder teniendo sesión iniciada con un mecánico aparecerá el siguiente mensaje:

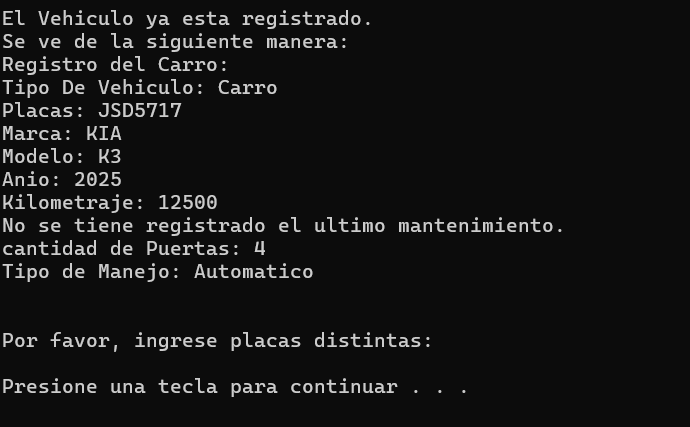
Y nos regresa al menú principal; ahora, accediendo con un administrador se nos despliega el siguiente menú:

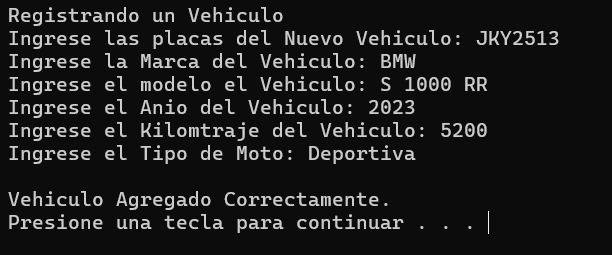
donde podemos elegir el tipo de vehículo a registrar, según el que seleccionemos, se nos pedirán ciertos datos, por ejemplo, elijamos un Carro:



Una vez validados los datos, se nos avisa que el vehículo ha sido agregado correctamente, por lo que la siguiente ventana que nos sale es la repetición:

Aquí podemos elegir si regresar al menú anterior o seguir registrando vehículos, intentemos ingresar ahora, una camioneta, pero, le pondremos las mismas placas que el vehículo que acabamos de registrar, como las placas es el identificar de cada vehículo, es decir, lo aquello con lo que GIVAM los diferencia, no pueden existir dos placas iguales para el correcto funcionamiento, así que si intentamos ingresar placas ya registradas, GIVAM nos avisa de aquello de la siguiente manera:



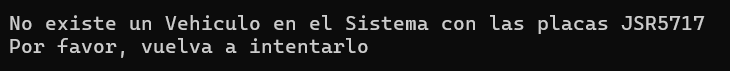
Nos muestra la información del vehículo que posee dichas placas y nos pide ingresar unas distintas (Nótese que como todavía, dicho vehículo no le hemos registrado mantenimientos, expresa esto mismo en la información), así que ingresando datos válidos, ingresaremos una camioneta y una moto:

Así podemos registrar cuantos vehículos sean necesarios.

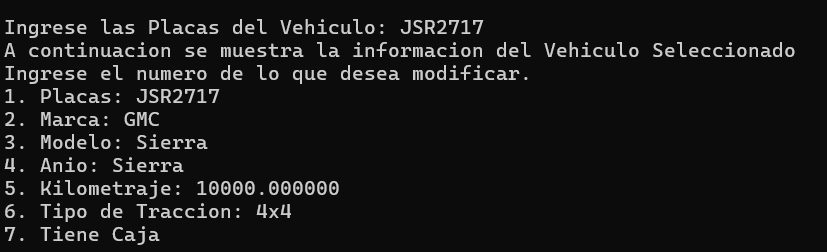
### Editar un Vehículo

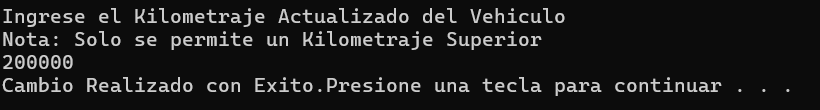
Ahora, siguiendo con la exploración del gestor de Vehículos, la segunda opción es la de editar un vehículo ya existente

En este menú podemos editar cualquiera de las especificaciones con la que lo hemos registrado, accedemos a ellos mediante las placas (Existe un mensaje de error y retorno si aún no hay vehículos registrados), placas no existentes se avisa de ello:



Ingresando las placas de la camioneta (un vehículo existente) se muestra lo siguiente:

Podemos ingresar el número de la característica a modificar (Cabe destacar que las placas deben ser distintas, sino pasará lo mismo que si intentamos registrar un vehículo con placas registradas: nos mostrará la información del vehículo registrado y regresa), modifiquemos el Kilometraje:

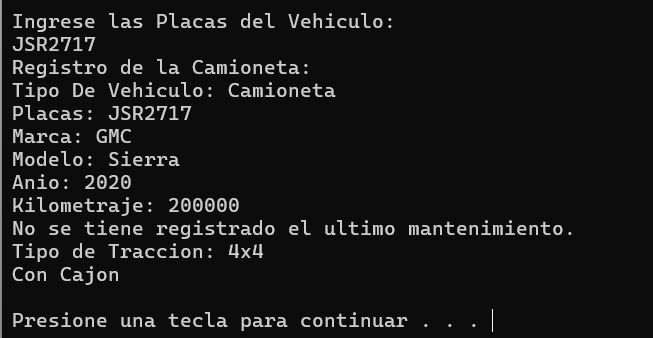
Nos avisa que el cambio se realizó exitosamente y regresamos al menú, luego nos pregunta si queremos realizar otra edición:



Y así cuanto necesite el administrados.

### Buscar un Vehículo

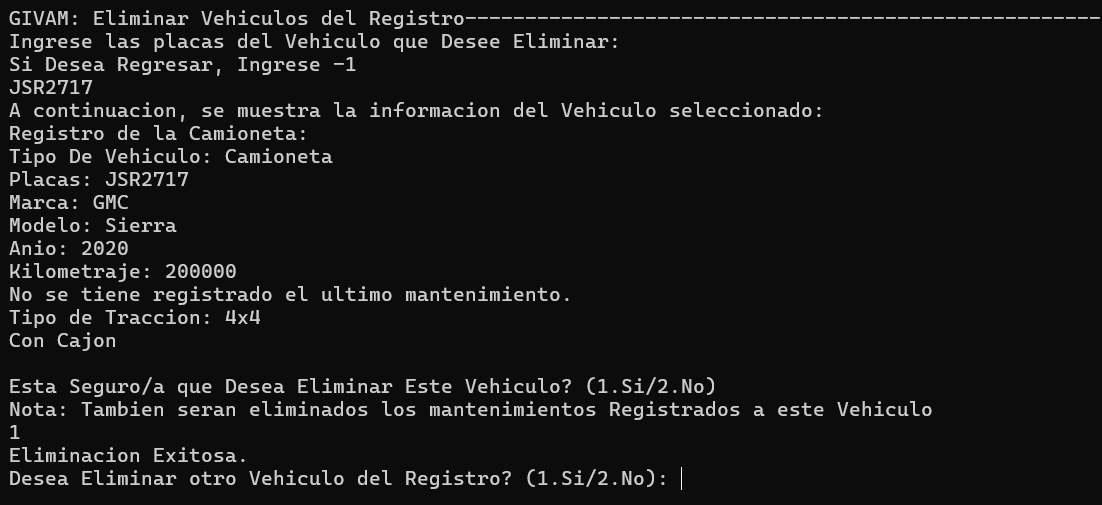
En esta opción simplemente nos pide las placas de un vehículo registrado y muestra su información (Una opción disponible para todo tipo de usuarios)

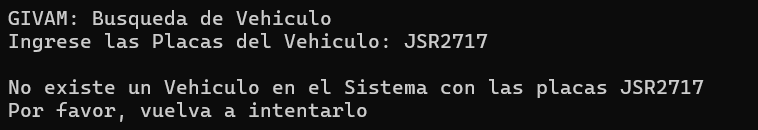


Nótese que el kilometraje se ha actualizado como lo hemos hecho en con anterioridad.

### Eliminar un Vehículo

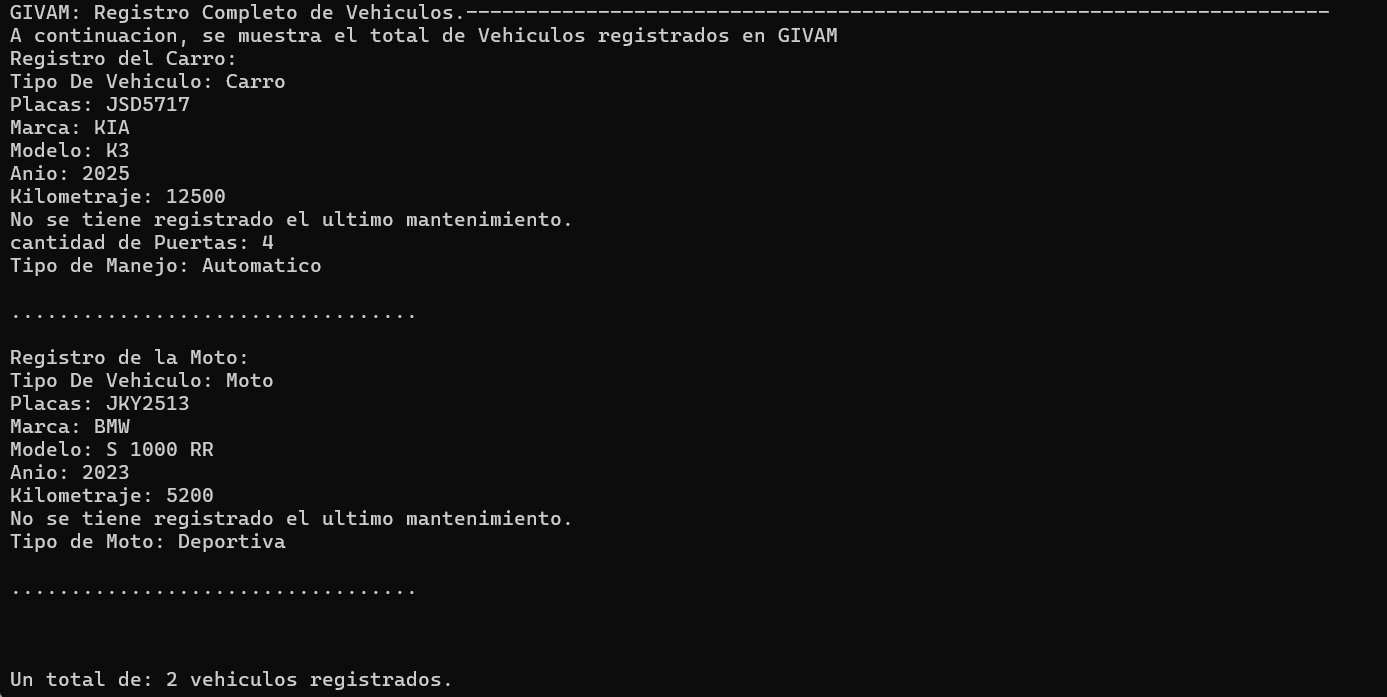
Se nos piden las placas de un Vehículo, nos muestra su información y nos pide una confirmación, después de ello, se nos avisa que la eliminación fue exitosa.

 Ahora ya no aparecerá más en los registros, si intentamos buscar sus placas ya no aparecerán registradas:



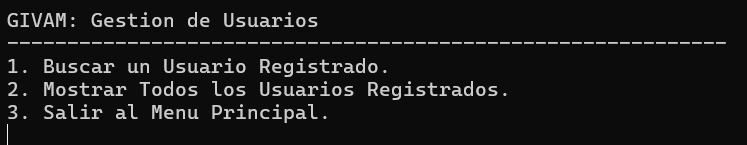
### Mostrar el Registro de Todos los Vehículos

La última opción del gestor de vehículos es la de mostrar todo el registro, al seleccionarla, nos imprime la información de todos los vehículos registrados:

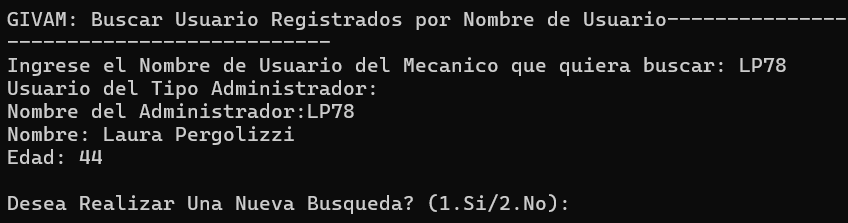
y se nos da el total de registros.

## Gestión de Usuarios

La segunda opción del menú principal es el gestor de usuarios, en él, se nos muestran estas dos opciones:

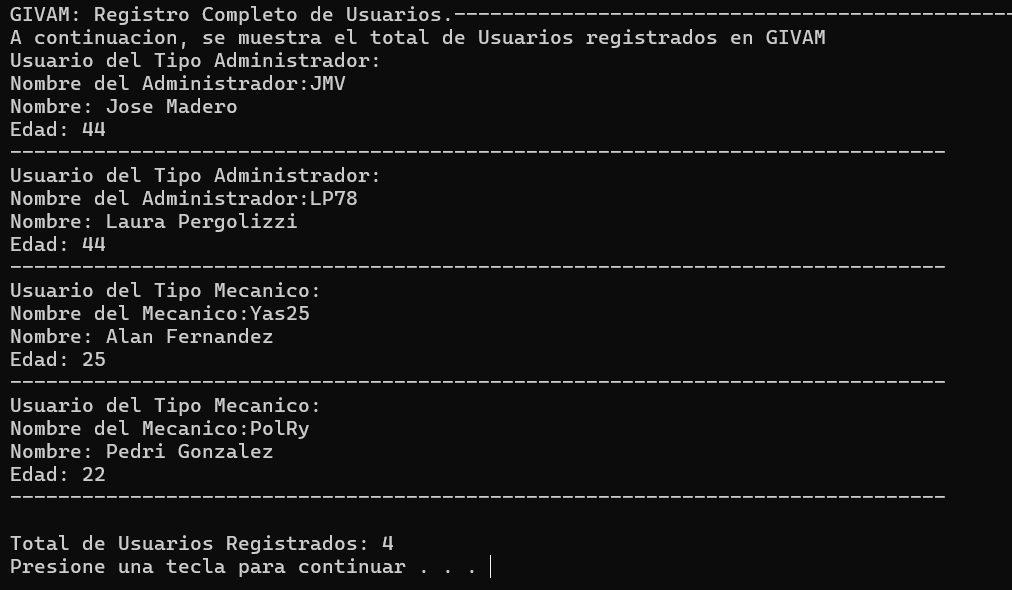


### Buscar un Usuario Registrado

La primera de estas, nos pide un nombre de usuario registrado y muestra la información de aquel:

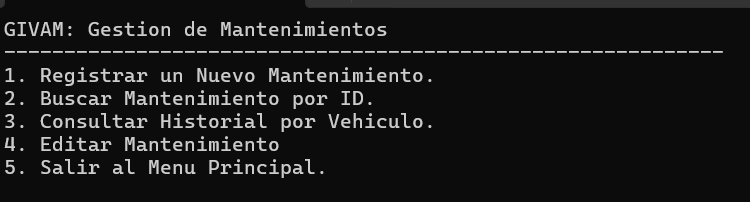
### Mostrar Registro de Usuarios Registrados

La segunda opción del gestor de usuarios, al igual que en los vehículos, imprime la información de todos los usuarios registrados y contando el total:

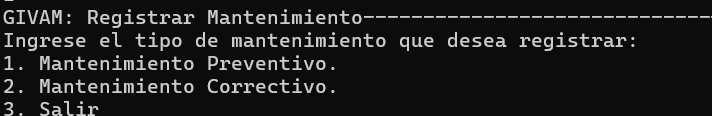


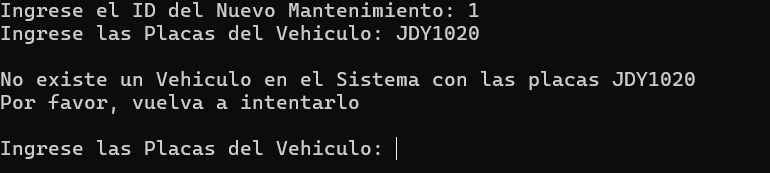
## Gestor de Mantenimientos

El gestor de mantenimientos nos presenta las siguientes opciones:

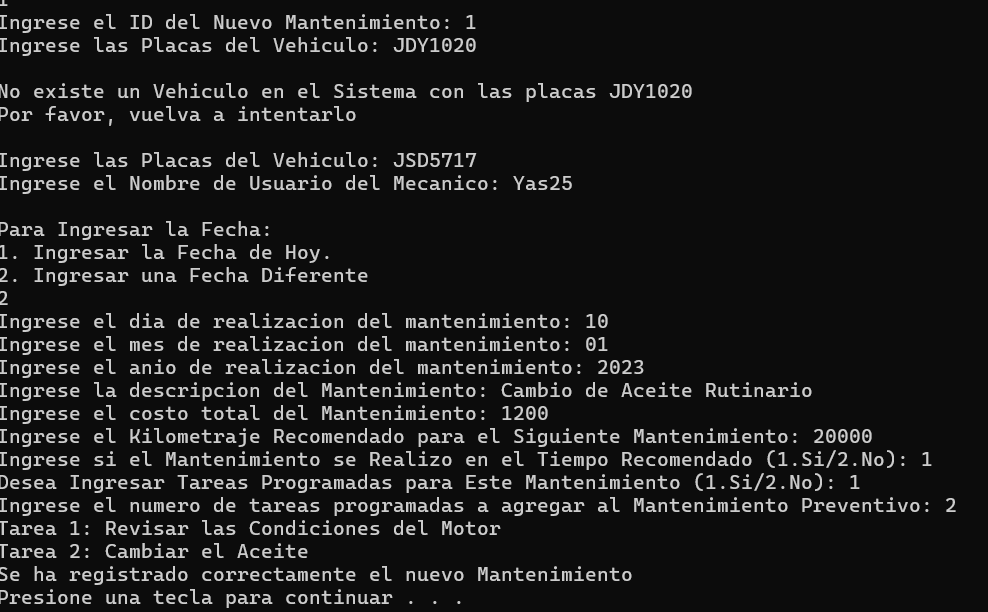
Cada uno es un homónimo a las del vehículo

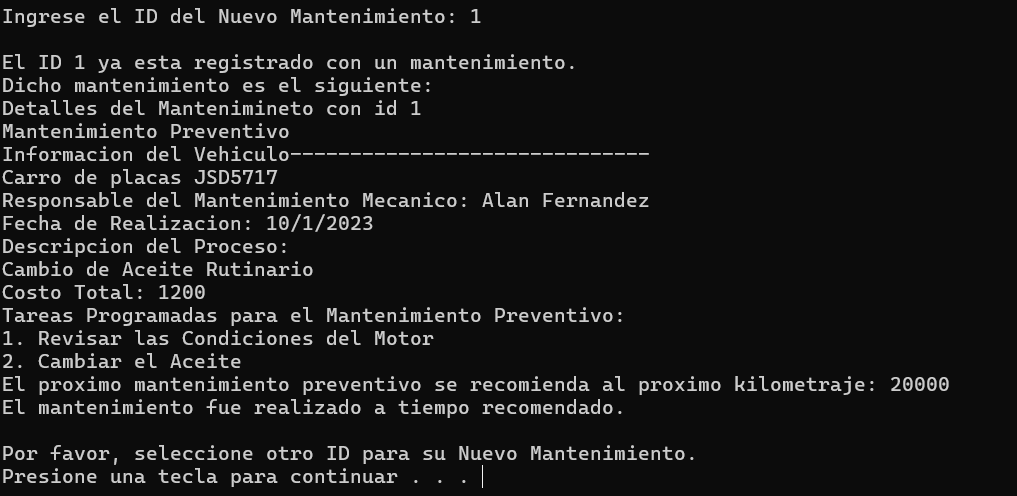
### Registrar un Nuevo Mantenimiento

Primero se nos pide el tipo de mantenimiento, si es uno preventivo (como lo podría ser un cambio de aceite, chequeo rutinario, revisión de presión de llantas, etc) o uno correctivo (si el vehículo ya presenta alguna falla consigo), cada uno recabando información propia, primero, agreguemos un Mantenimiento Preventivo:

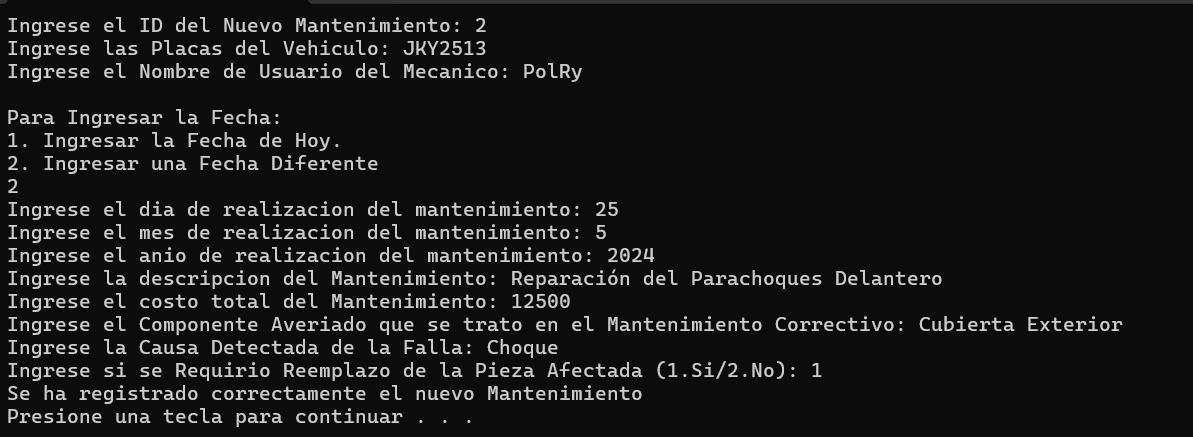
Para ambos mantenimientos, se nos pedirá un ID numérico y las placas del vehículo al cuál se le aplican.

Estas placas deben estar registradas, caso contrario se nos avisa y nos pide unas existentes (como nota, si no hay vehículos registrados, ni siquiera podremos acceder a registrar un mantenimiento, ya que no habrá vehículos a los cuales asignárselas)

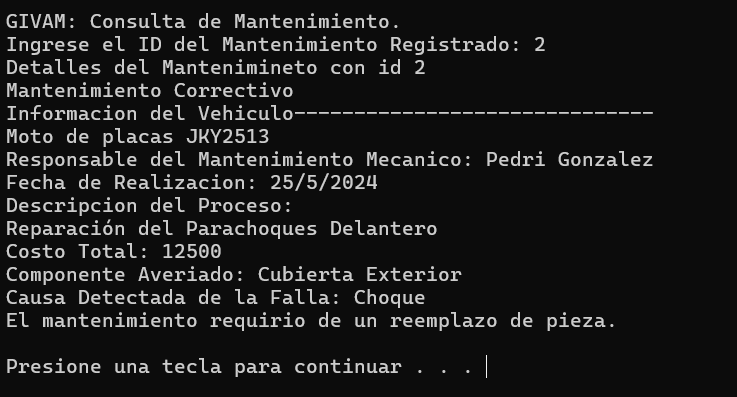
Al igual que los vehículos, los mantenimientos se anexan a un mecánico, por lo que es necesario ingresar el nombre de usuario de uno existente, los demás son datos propios del mantenimiento. Una vez ingresados, se nos avisa que el mantenimiento fue registrado con éxito. Para seguir con el ejemplo, agregaré un Mantenimiento Correctivo.

Si intentamos ingresar un ID ocupado, nos aparece lo siguiente:

Así, que registrando un Mantenimiento Correctivo con un ID disponible:

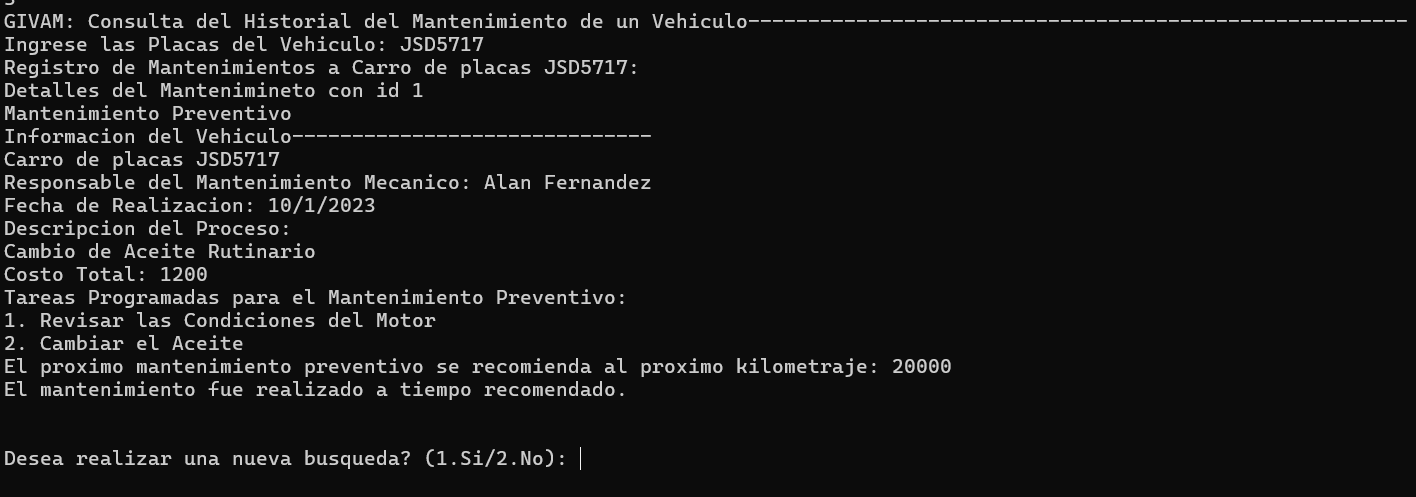


### Buscar Mantenimiento por ID

Se pide un ID de mantenimiento registrado y se muestra la información si es que este existe:

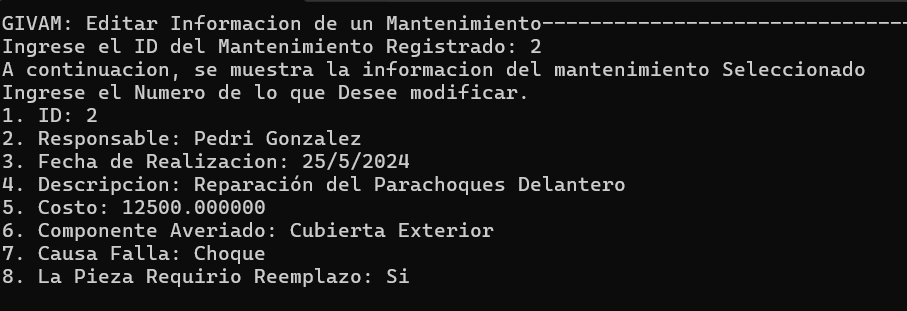
### Consultar Historial por Vehículo

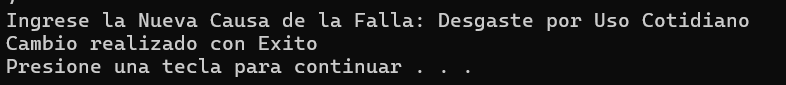
Aquí podemos, mediante las placas de un vehículo, imprimir todos los mantenimientos que se le hayan realizado:

En caso de tener múltiples, se van mostrando todos consecutivamente

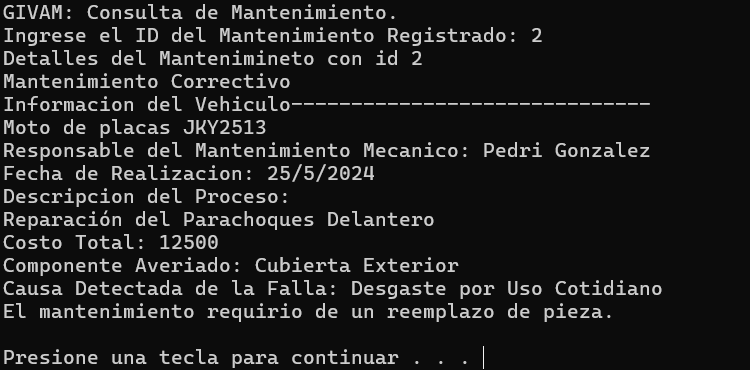
### Editar un Mantenimiento

Al igual que en los vehículos, podemos editar la información ya registrada de un mantenimiento siendo administradores del sistema, ingresando un ID existente:

y posteriormente, eligiendo qué cosa cambiaremos:

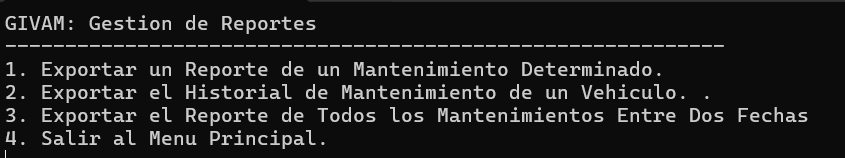


Los cambios se ven registrados si volvemos a buscar el mismo mantenimiento por ID:

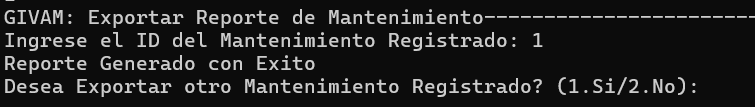


## Reportes

La Cuarta opción del menú principal está dedicada a la creación y exportación de reportes de mantenimientos, al seleccionarla, aparece el siguiente submenú:

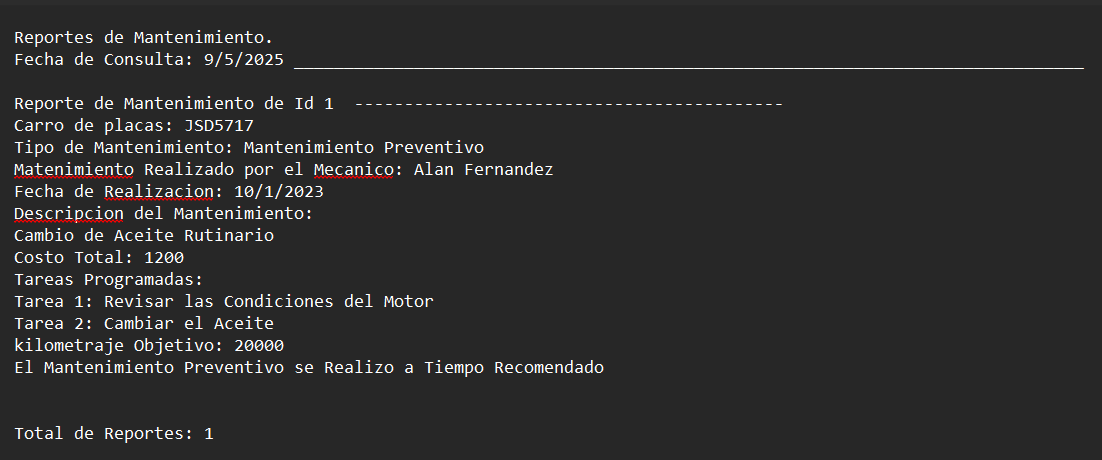
Exploremos la primera opción:

### Exportar un Reporte Determinado

Se nos pedirá un ID de reporte registrado, y al seleccionarlo, se nos dirá que el reporte fue generado con éxito, si nos fijamos en la carpeta donde nuestro programa se ejecuta tendremos un archivo como este:

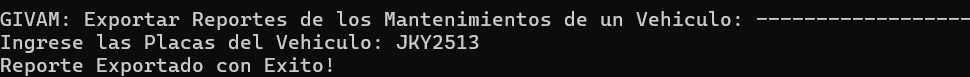


Un archivo .txt con un título característico, y esto contiene en su interior:

Escrito en texto plano, están las características de nuestro reporte con la información relevante de aquello.

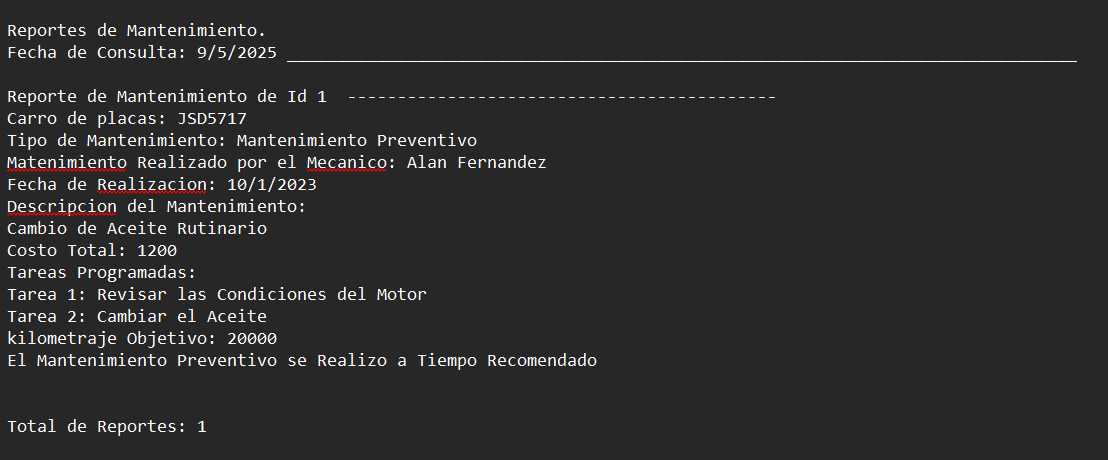
### Exportar Reportes de un Vehículo

Esta función es similar a la anterior, pero en lugar de exportar un mantenimiento único por ID, exporta todo el historial de mantenimientos de un vehículo ingresando sus placas:



Generando un archivo .txt como el siguiente:

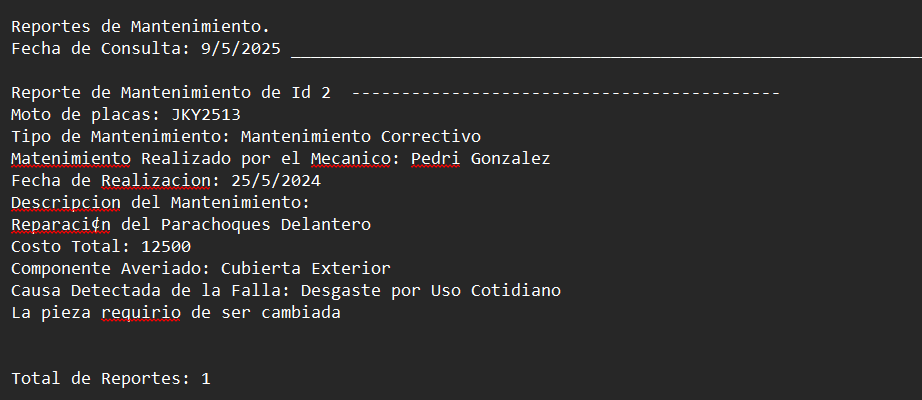




### Exportar Mantenimientos Entre Fechas

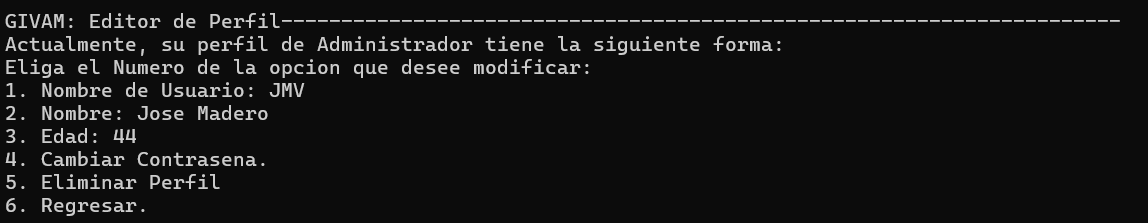
Aquí, el filtro para seleccionar qué mantenimientos registrará serán dos fechas dadas, ahí se incluirán todos los mantenimientos entre dichas Fechas:

Generando un archivo .txt como el siguiente:

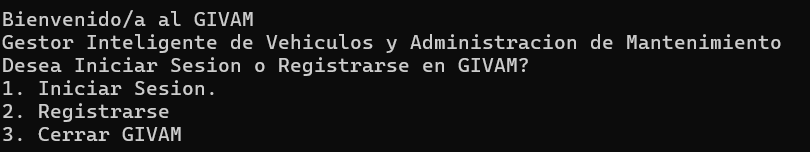


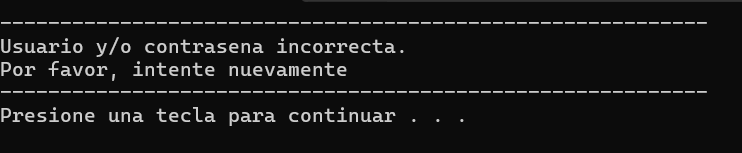
## Configuración de su Cuenta

Esta opción del menú principal nos dirige a la siguiente ventana:

nos muestra los datos sobre los cuales está iniciada nuestra sesión y aquí podemos editar cualquiera de ellos o eliminar el perfil, para demostrar más funcionalidades y ver qué pasa, seleccionemos la 5ta, eliminar el perfil:

Se nos pide una confirmación, y al aceptarla nos avisa que la eliminación fue exitosa y nos redirige a la primera pantalla de GIVAM:

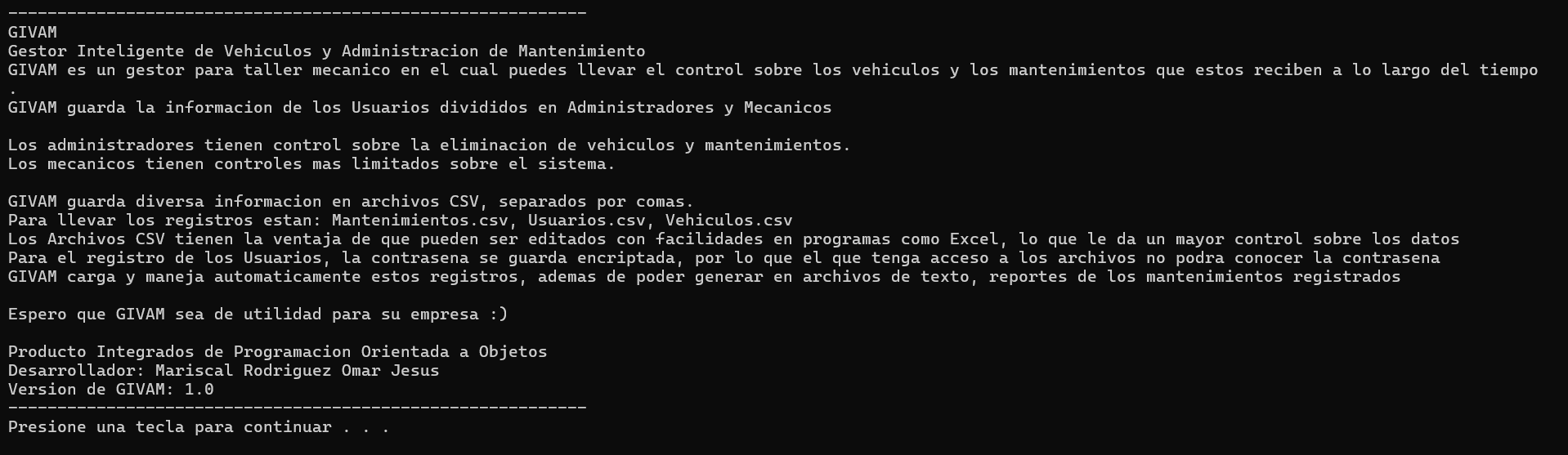
Y si intentamos acceder con las mismas credenciales pasará esto:

Por lo que tendremos que, o registrar otro perfil, o bien, acceder con otra sesión.

Así mismo, podemos editar cualquier otro dato que se nos presente.

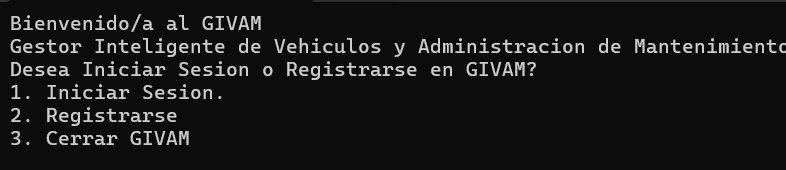
## Información de GIVAM

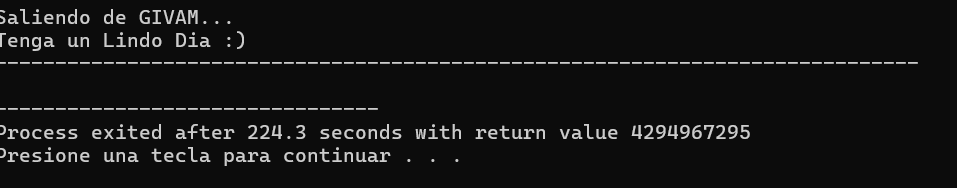
Esta penúltima opción del menú principal nos arroja información relevante sobre el funcionamiento de GIVAM y sobre su versión:

Útil para el control de versiones si GIVAM se llega a actualizar más adelante.

## Cerrar Sesión

Nos devuelve a la primera pantalla de GIVAM, cerrando la sesión

Y si seleccionamos Cerrar GIVAM:



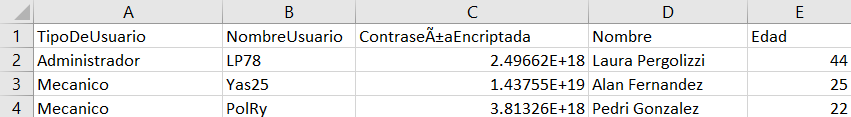
## Control de Almacenamiento

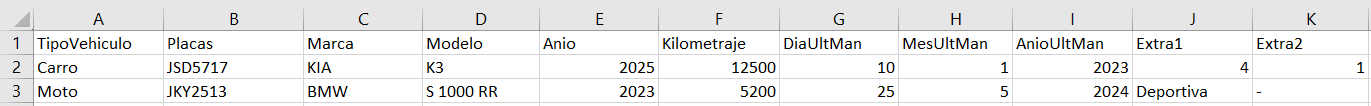
GIVAM está programado para sostener sus datos, aún si el programa cierra, para ello, todos estos registros que hemos estado haciendo se almacenan en archivos .csv dentro de la misma carpeta donde GIVAM se ejecuta

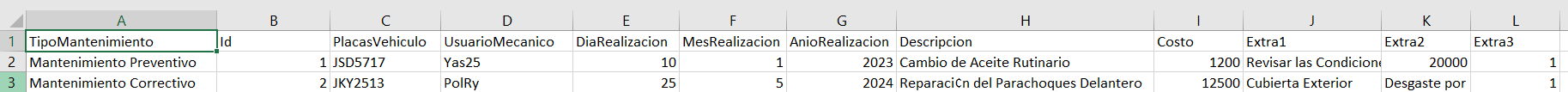


Estos archivos contienen la información que hemos registrado, al ser formato .csv, se hace sumamente fácil manejarla y visualizarla con aplicaciones externas como Excel, por lo que editando esto GIVAM en automático carga los datos aquí registrados, así que sirve como una manera de importación si una empresa adquiere GIVAM y quiere ingresar todo su catálogo, no haciéndolo uno por uno a través de la consola, sino más fácilmente con Excel

El almacenamiento tiene un look como el siguiente:

En este caso particular de los usuarios, la contraseña se almacena encriptada mediante un Hash, por lo que aunque alguien acceda al almacenamiento .csv, la contraseña no válida, GIVAM no desencripta la contraseña, sino lo que hace es encriptar también el intento de contraseña y compararla con este hash, por lo que la contraseña real no se almacena en ningún momento.

Para Vehículos el archivo .csv se ve como:

Y para los Mantenimientos:

Su interpretación y carga es gestionada por el mismo GIVAM.

De esta manera, hacemos que el sistema tenga su propio método de almacenamiento de datos para que el programa sea consecuente y toda su información no sea eliminada en cuanto cierra.

Completando así, los puntos que GIVAM pretendía solucionar como un sistema integral para la gestión de vehículos, usuarios y mantenimientos de un taller automotriz.

# Conclusiones de la Actividad

La realización del sistema GIVAM representó un buen reto para integrar todos los conocimientos y el paradigma recabado en el curso, en la práctica he aprendido muchas cosas, empezando por darle el valor que merecen a los diagramas UML, en un inicio parecen solo esquemas que retrasan el empezar a programar lo que más o menos se tiene en mente, pero con un problema más grande como fue GIVAM, el tener un modelo sólido sobre lo que GIVAM debe ser y qué camino se debe tomar agiliza mucho el proceso de la programación, sabiendo qué queremos hacer es mucho más fácil hacer el código más eficiente en términos de líneas de código y en cuestiones de tiempo; hace que no tengamos que retroceder una y otra vez para añadir más clases, más métodos o ver qué si queremos añadir una función ahora tenemos que modificar muchas más cosas. Además, puedo entender que su valor más allá incluso en el trabajo colaborativo, si cada persona debe trabajar en un módulo, una clase o ciertas funcionalidades, no es la mejor idea esperar que un programador acabe su módulo para empezar a trabajar con el mío porque requiero de un resultado de su función, sino que el plantear de antemano qué debe retornar cada función permite tener valores de prueba y trabajar a la vez en funciones que, dentro del código dependen unas de otras; una herramienta de importancia fundamental que al inicio del curso no podía valorar.

Otro gran aprendizaje que tuve en la realización de esta actividad integradora es procurar, dentro de la fase de desarrollo tener funciones específicas que cumplan tareas concretas, esto es bastante útil para mantener el código legible, si ocurre un error identificarlo más fácilmente y hacer el código más expandible; por ejemplo, dentro de mi código GIVAM se encarga de coordinar las acciones de cada una de las clases que la componen, pero GIVAM recibe datos que ya deben estar filtrados, y el incluir dentro de él también los menús o el filtro de datos rompe con el principio de responsabilidad única y es una mala praxis de programación, por lo que creé otra clase llamada Menú encargada de presentar menús y submenús y de filtrar los datos que GIVAM recibe, así identificando más fácil los errores que surgieron y haciendo más mantenible, legible y con mejor presentación el código final.

Cuestión de polimorfismo, es increíble lo sumamente útil que es, también en conjunto con las clases abstractas, no veía la necesidad de incluir métodos abstractos que no definieran su código, pero el tenerlos ahí y poder aplicar polimorfismo dinámico a sus clases padres fue algo de gran provecho en todo el código; valoré también más el tema del encapsulamiento, creo que fue el pilar de POO que más cuestioné en un inicio, pensaba por qué habría que limitar el acceso a los datos ¿no era más cómodo tenerlos a la mano y dejar de utilizar getter’s y setter’s que son funciones simples que podrían desaparecer sin el encapsulamiento? Con GIVAM aprendí que existen funciones o datos que como programador, sólo quiero que se ejecuten desde un punto específico del código, acceder a ellos desde otras partes puede tener consecuencias fatales.

Un refuerzo de aprendizaje que tuve también fue el manejo de memoria dinámica, si bien, el curso no va centrado únicamente en C++ sino que en el paradigma de la Programación Orientada a Objetos, trabajar con C++ y su libertad para el manejo de la memoria es una responsabilidad destacada y crucial, y estar utilizando apuntadores, vectores de apuntadores y gestionar este apartado sin tener fugas fue enriquecedor.

En conclusión, el desarrollo de GIVAM fortaleció en gran manera desde mi parte práctica hasta mi parte conceptual de los pilares de este paradigma y me hizo valorar mucho más cada uno de los por qué’s de cada concepto y la importancia de los modelos en esta profesión.