



Pablo Zuil 1 DAW

Alias : kkarrasmil80

Sección de contenidos

1. Requisitos

- a. Requisitos funcionales
- b. Requisitos no funcionales

2. Diagrama de Entidad Relación (E/R)

- a. Explicación de diseño
- b. Explicación de las relaciones
- c. Herramientas utilizadas

3. Diagrama de clases (modelo de negocio)

- a. Explicación de diseño
- b. Herramientas utilizadas
- c. Explicación de relaciones

4. Diagrama de casos de uso

- a. Explicación de los actores que intervienen
- b. Explicación de cada caso de uso

5. Diagrama de Navegación

- a. Explicación del flujo a seguir del cliente

6. Trello

- a. Tareas completadas y no completadas

7. Informe de cobertura y test

- a. Cobertura total del código

8. Estimación de costes

9. Conclusiones

1. - REQUISITOS

1.1- Requisitos funcionales

- a. Realizar las operaciones CRUD(create, read, update y delete) para poder gestionar vehículos y citas
- b. Poder diferenciar entre vehículos (Vehículo tipo motor, eléctrico y público)
- c. Validar los datos de los Vehículos para que puedan pasar las pruebas
- d. Validar los datos introducidos en las citas para que se creen correctamente
- e. Implementar un inicio de sesión sólido, que se conecte con la BBDD
- f. Permitir la importación en CSV y JSON a la vez que su exportación en los mismos formatos
- g. Permitir generar informe de la prueba realizada al vehículo al ser completada
- h. Crear las viseras que gestionen de manera única y responsable cada parte del programa y de la UI de javaFx

1.2. - Requisitos no funcionales

- a. Todos los archivos relacionados con los clientes deberán de ser testeados
- b. Todos los archivos relacionados con los vehículos deberán de ser testeados
- c. Todos los archivos relacionados con las citas deberán de ser testeados
- d. Debe de poderse ejecutar un .jar en la terminal del IDE o del sistema operativo
- e. Trabajar con las herramientas de control de versiones GitFlow y GitHub
- f. Comentar el código para una mayor legibilidad y entendimiento
- g. Documentar el código con el plugin de Dokka

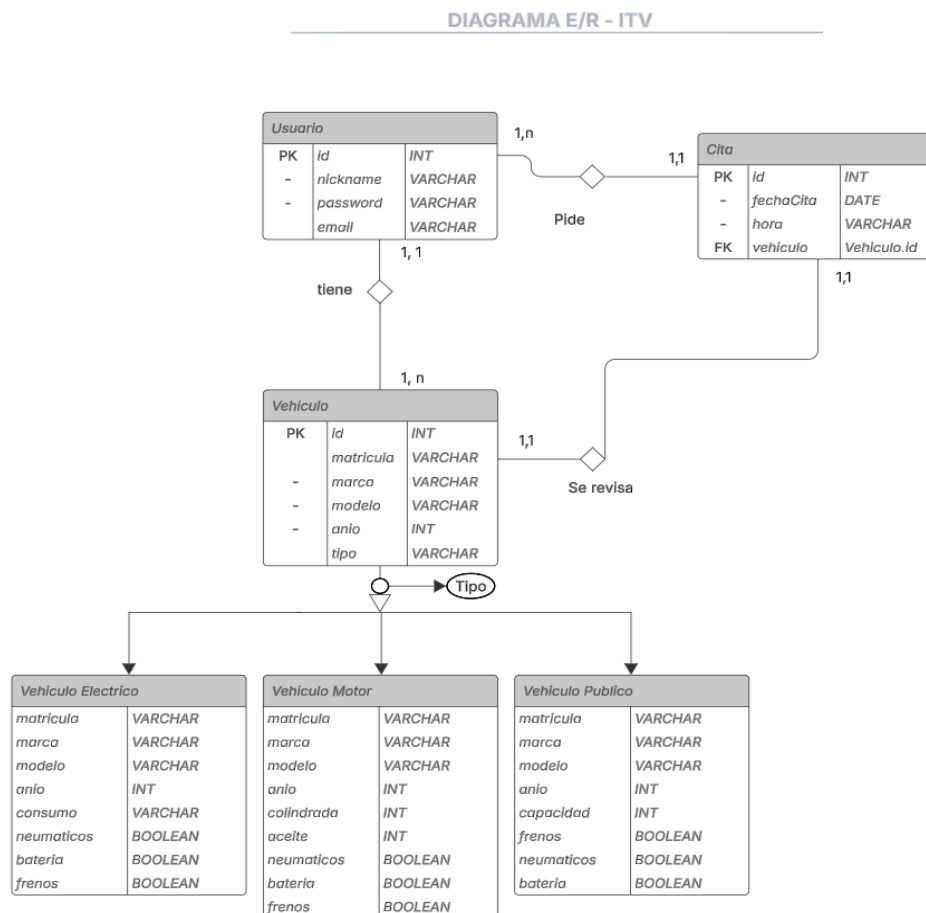
3. - Diagrama Entidad relación (E/R)

3.1 - Explicación de diseño

Tenemos 6 tablas principales, La tabla usuario que tiene como clave primaria un identificador único.

La tabla citas que tiene como clave primaria un identificador único, el resto de atributos que la asemejan y un clave foránea que referencia al identificador único de un coche, ya que en una cita, hay un coche, si no no habria nada que validar.

También tenemos la tabla Vehiculo, que tiene como clave primaria un identificador del que hemos hablado anteriormente. Tiene una jerarquia que la divide en tres subclases Vehiculo Motor, Vehiculo Electrico y Vehiculo Publico (refiriendose a un vehiculo de transporte publico) los cuales tienen sus parametros propias para que las pruebas en cada uno sean mas “unicas”



3.2 - Explicación de relaciones

Como podemos observar, en el diagrama existen relaciones entre las tablas y a continuación las explicaré paso a paso.

La tabla citas se relaciona con el Usuario/Cliente ¿A qué se debe? Esto se debe a que el usuario para poder validar su coche necesita pedir una cita. Un cliente puede pedir 1 y 1 sola cita cada 6 meses (por ejemplo) pero las citas pueden ser pedidas por varios usuarios.

A su vez un usuario necesita un coche para poder validar, por ello llegamos a la conclusión de que un propietario tiene 1 o varios vehículos pero todos ellos pertenecen a el mismo.

Por último la relación de citas y coches, una cita depende totalmente de un coche por lo que es una entidad débil, ya que sin coches, no habría citas. Una cita revisa solo 1 solo coche a la vez y el coche es revisado en 1 sola cita.

3.3 - Herramientas utilizadas

Para crear el diagrama E/R de Base de Datos he utilizado Lucidchart debido a su facil manejo, sencillez y presentación.

4. - Diagrama de Clases (UML)

4.1 - Explicación de diseño

En este diagrama de clases tenemos representadas las entidades principales que conforman el sistema GESTOR-ITV desde una perspectiva orientada a objetos. Se identifican clases como Usuario, Cita y una clase abstracta Vehiculo, que a su vez se especializa en tres subclases: VehiculoMotor, VehiculoElectrico y VehiculoPublico.

Estas clases definen sus atributos y métodos clave, necesarios para la gestión del sistema. Por ejemplo, Usuario tiene métodos como iniciarSesion() y cerrarSesion(), mientras que Cita incluye funciones como crearCita() o validar().

Además, el diseño aplica principios SOLID, como el uso de interfaces (ValidatorCliente, ValidatorVehiculo, ValidatorCita) que encapsulan la lógica de validación específica para cada clase. Esto permite mantener el sistema desacoplado y más fácil de mantener y escalar.

También es importante destacar que la clase Vehiculo actúa como abstracta porque representa una generalización de lo que significa un coche dentro del sistema. Las clases hijas heredan de ella e implementan su propia lógica en métodos como validar() o importarCoche().

4.2 - Explicación de relaciones

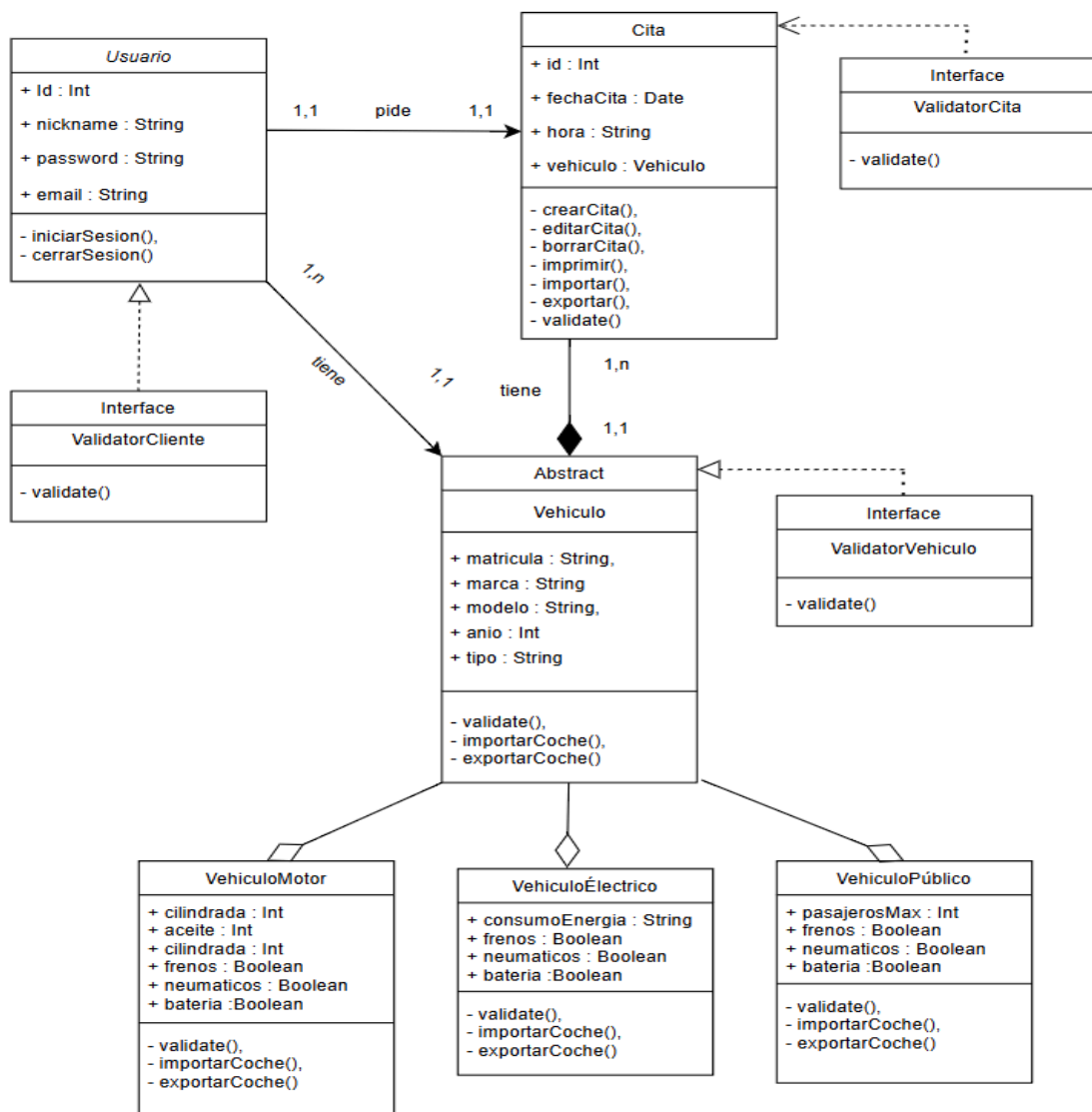
Las relaciones reflejadas en este diagrama están basadas en la lógica de negocio:

- Un Usuario puede tener uno o varios Vehiculo, pero un Vehiculo pertenece únicamente a un Usuario. Esto representa la relación de propiedad de los vehículos.
- Un Usuario puede pedir una Cita, mientras que la Cita está relacionada con un único Vehiculo. La cita es necesaria para que el vehículo pueda ser validado correctamente por el sistema.
- Las subclases de Vehiculo están relacionadas mediante herencia con la clase abstracta Vehiculo. Cada tipo de vehículo tiene atributos específicos, como cilindrada o consumoEnergia, que determinan qué pruebas se deben realizar en cada revisión técnica.
- Las interfaces de validación están conectadas con las clases que requieren dicha funcionalidad, asegurando que cada una implemente

su propia lógica de validación, acorde a su naturaleza.

4.3 - Herramientas utilizadas

El diagrama de clases UML ha sido elaborado utilizando draw.io, por su facilidad de uso, diseño limpio y herramientas intuitivas que permiten representar de manera clara y profesional las relaciones y estructuras de una aplicación basada en clases.



5. - Diagrama de Casos de Uso del Sistema

5.1 - Explicación de diseño

Tenemos dos actores principales en nuestro sistema: el Cliente y el Validador. El sistema, en su conjunto, se denomina "Gestor de ITV".

El Cliente interactúa directamente con el sistema para realizar acciones relacionadas con la gestión de su vehículo y la solicitud de citas. Las principales funcionalidades a las que tiene acceso el Cliente son: "Iniciar Sesión", "Consultar vehículos", "Validar vehículos" y "Entrar al panel de citas".

Dentro del caso de uso "Consultar vehículos", se desprenden varias acciones que pueden ser realizadas por el cliente: "Importar Vehículos", "Exportar Vehículos", "Crear, editar y eliminar vehículos", "Filtrar Vehículos" e "Imprimir PDF o HTML". Todas estas funcionalidades, a su vez, "incluyen" el caso de uso "Validador", lo que sugiere que estas acciones requieren una validación o procesamiento por parte del "Validador".

El caso de uso "Validar vehículos" "extiende" la funcionalidad de "Imprimir PDF o HTML", lo que indica que para validar un vehículo, podría ser necesario generar un documento.

Finalmente, el caso de uso "Entrar al panel de citas" lleva a la acción de "Crear, editar y eliminar citas", que también "incluye" el caso de uso "Validador", implicando que la gestión de citas también requiere una validación.

El Validador es el otro actor, y su rol principal es interactuar con el caso de uso "Validador", que es incluido por varias de las funcionalidades principales del Cliente. Esto implica que el Validador es responsable de aprobar o procesar las operaciones que realiza el Cliente relacionadas con vehículos y citas.

5.2 - Explicación de relaciones

Como podemos observar, en el diagrama existen relaciones entre los actores y los casos de uso, y a continuación las explicaré paso a paso.

El Cliente se relaciona directamente con los casos de uso "Iniciar Sesión", "Consultar vehículos", "Validar vehículos" y "Entrar al panel de citas". Esto se debe a que el cliente es quien inicia estas acciones en el sistema.

El caso de uso "Consultar vehículos" se relaciona con "Importar Vehículos", "Exportar Vehículos", "Crear, editar y eliminar vehículos", "Filtrar Vehículos" e "Imprimir PDF o HTML" mediante una relación de dependencia, lo que significa que estas acciones son sub-funcionalidades de la consulta de vehículos.

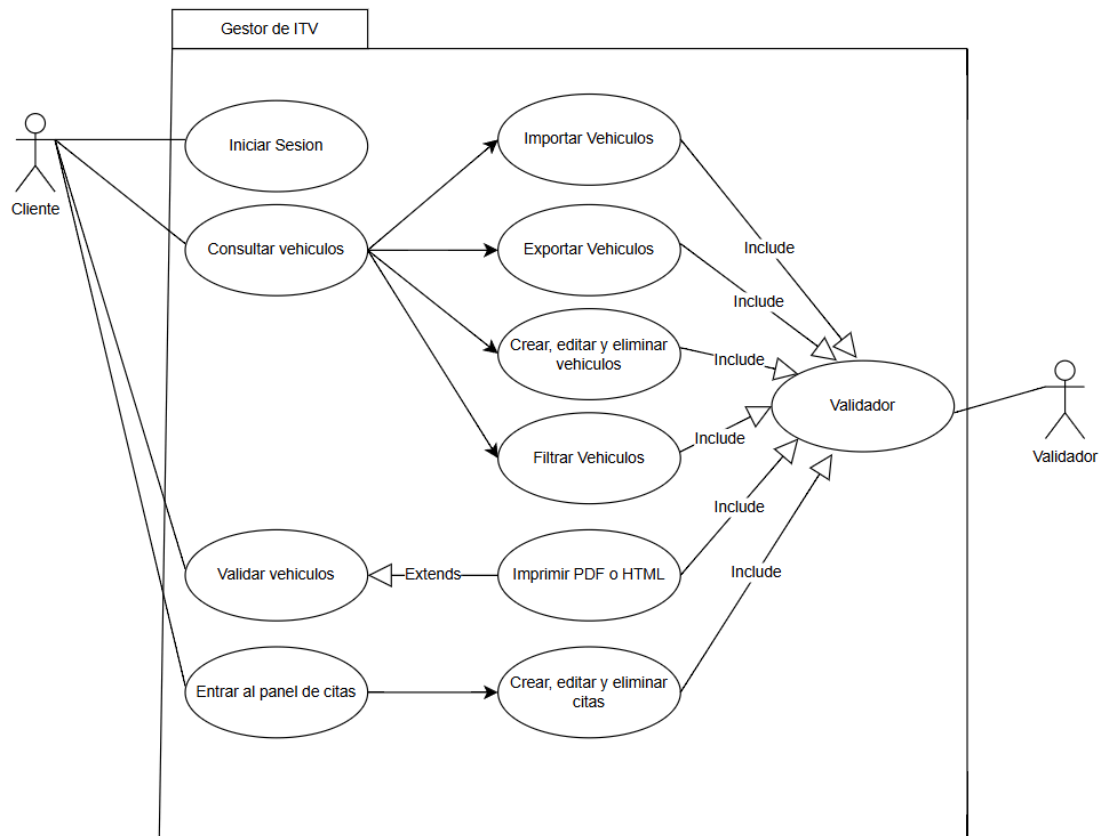
Todas estas sub-funcionalidades, junto con "Crear, editar y eliminar citas", "incluyen" el caso de uso "Validador". Esto indica que para que estas operaciones se completen, se requiere la intervención o aprobación del Validador. Es decir, sin la acción del Validador, estas operaciones no pueden finalizar.

El caso de uso "Validar vehículos" "extiende" la funcionalidad de "Imprimir PDF o HTML". Esto significa que la impresión de un documento (PDF o HTML) es una funcionalidad opcional que puede activarse durante el proceso de validación de vehículos.

Por último, el caso de uso "Entrar al panel de citas" conduce a la acción de "Crear, editar y eliminar citas", lo que implica que el acceso al panel es el prerequisite para gestionar las citas.

3.3 - Herramientas utilizadas

Para crear el diagrama de Casos de Uso del Sistema he utilizado draw.io debido a su facilidad de manejo, sencillez y presentación.



6. - Diagrama de Navegación del Sistema

6.1 - Explicación de diseño

Tenemos un flujo de navegación que comienza con la "Pantalla de carga", que es el punto de entrada inicial al sistema. Desde aquí, el usuario es dirigido a la pantalla de "Inicio de sesión", donde se le solicita que introduzca sus credenciales.

Tras el "Inicio de sesión", existen dos posibles caminos: si el "Login correcto", el usuario accede a la "Vista usuario"; si el "Login incorrecto", se regresa a la pantalla de "Inicio de sesión" para reintentar el acceso.

Una vez en la "Vista usuario", el usuario puede optar por "Cerrar sesión", lo que lo devuelve a la pantalla de "Inicio de sesión", o proceder a la "Gestión del usuario", que es el centro de las funcionalidades principales del sistema.

Desde la "Gestión del usuario", el sistema se bifurca en dos ramas principales de funcionalidad: el "Panel del vehículo" y el "Panel de citas".

Desde el "Panel del vehículo", el usuario puede "Importar información del vehículo", lo que a su vez le permite "Validar vehículo" y, posteriormente, "Exportar PDF, HTML, JSON y CSV". También existe la opción de "Volver al inicio" desde el "Panel del vehículo", lo que regresaría a la "Gestión del usuario".

Desde el "Panel de citas", el usuario puede acceder a la "Consulta de vistas", y desde esta, realizar una "Operación de cita para la ITV". De igual manera, desde la "Consulta de vistas", existe la opción de "Volver al inicio", regresando a la "Gestión del usuario".

6.2 - Explicación de relaciones

Como podemos observar, en el diagrama existen relaciones que indican el flujo de navegación entre las diferentes pantallas y funcionalidades del sistema, y a continuación las explicaré paso a paso.

La "Pantalla de carga" es la primera en mostrarse, llevando directamente al "Inicio de sesión". Esta es una relación de inicio de flujo.

Desde el "Inicio de sesión", la relación con la "Vista usuario" se activa solo si el "Login correcto". Si el "Login incorrecto", la relación de retorno hacia el mismo "Inicio de sesión" se activa, indicando que el usuario debe reintentar.

Una vez en la "Vista usuario", el usuario tiene dos opciones de navegación: "Cerrar sesión", lo cual lo devuelve al "Inicio de sesión", o avanzar a la "Gestión del usuario", que es el siguiente nivel de acceso a funcionalidades.

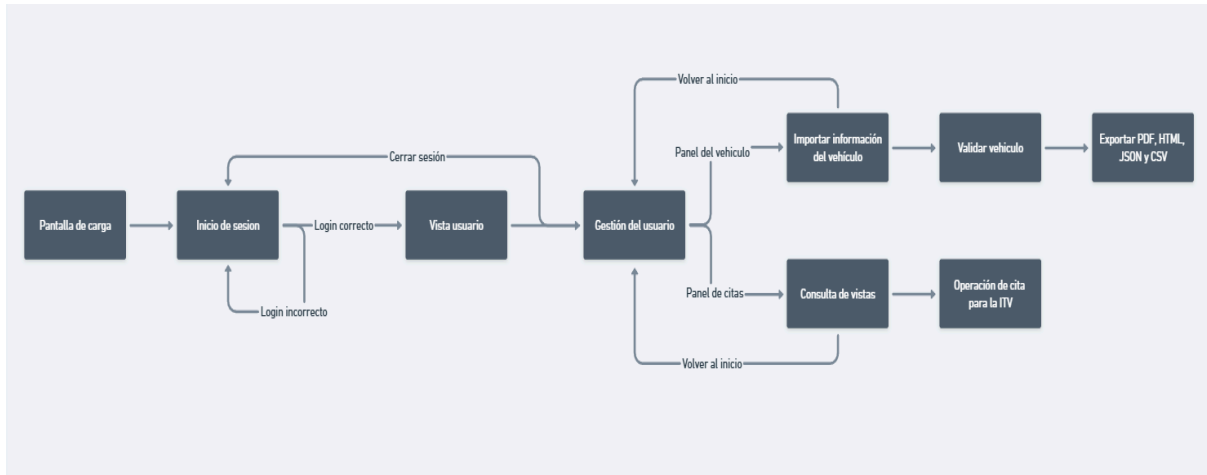
La "Gestión del usuario" es un punto de ramificación clave. Desde aquí, el usuario puede navegar hacia el "Panel del vehículo" o el "Panel de citas". Esto representa una relación de selección de área funcional.

Desde el "Panel del vehículo", la navegación continúa hacia "Importar información del vehículo". Esta acción, a su vez, habilita la navegación hacia "Validar vehículo". Una vez validado, la relación se establece con "Exportar PDF, HTML, JSON y CSV", indicando que la exportación es el paso final después de la validación. La relación "Volver al inicio" desde el "Panel del vehículo" indica que se puede regresar a la "Gestión del usuario".

De manera similar, desde el "Panel de citas", la navegación se dirige a la "Consulta de vistas", y desde esta, a la "Operación de cita para la ITV". La relación "Volver al inicio" desde la "Consulta de vistas" indica un retorno a la "Gestión del usuario".

6.3 - Herramientas utilizadas

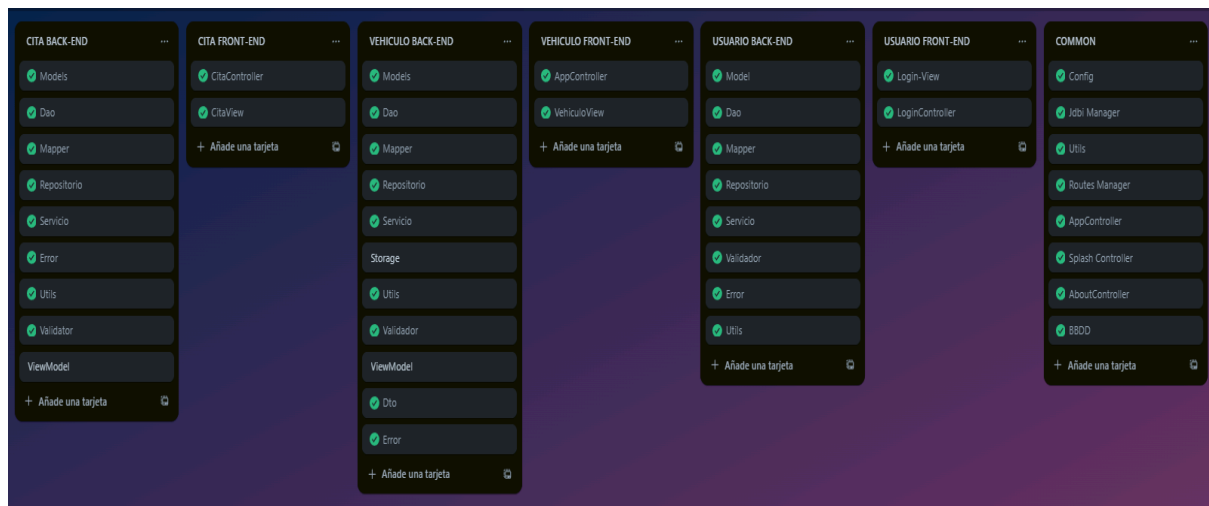
Para crear el diagrama de Navegación del Sistema he utilizado una herramienta de diseño de flujos llamada Whismical, debido a su facilidad de manejo, sencillez y presentación.



6. - TRELLO

6.1 - Tareas completadas y no completadas

El proyecto en su mayoría tiene casi todo implementado, pero faltan algunos detalles clave como el correcto uso del viewModel de citas en el controlador de Citas, el uso de imágenes para poder diferenciar vehículos de otros (parte del storage) que a su vez se conecta con el viewModel de vehiculos por lo que tampoco esta implementado del todo



6.2 - Cambios de diseño de ultima hora

En las funciones añadir, editar y eliminar (editar no esta implementada en Vehiculo), los métodos no respondían del todo bien con la interfaz de usuario ya sea por fallo en el diseño de la aplicación o del programa en si por lo que he utilizado pestañas para crear vehiculos y citas emergentes en vez de crear una vista aparte y conectar las funciones del servicio (save, update y delete), al crearse se añaden a la tableView de su escenario correspondiente. Adjunto la documentación de javafx que he encontrado sobre la decisión.

Constructor Summary

Constructors

Constructor	Description
<code>TextInputDialog()</code>	Creates a new <code>TextInputDialog</code> without a default value entered into the dialog <code>TextField</code> .
<code>TextInputDialog(String[®] defaultValue)</code>	Creates a new <code>TextInputDialog</code> with the default value entered into the dialog <code>TextField</code> .

Method Details

getEditor

```
public final TextField getEditor()
```

Returns the `TextField` used within this dialog.

Returns:

the `TextField` used within this dialog

En caso de duda, consulta :

<https://openjfx.io/javadoc/24/javafx.controls/javafx/scene/control/TextInputDialog.html>

En JavaFX, `TextInputDialog` es una clase de diálogo preconstruida que te permite mostrar una ventana emergente al usuario para que introduzca texto. Es una forma sencilla y eficaz de solicitar una entrada de texto rápida sin tener que diseñar un formulario completo o una nueva ventana desde cero.





























7. - INFORME Y COBERTURA DE TEST

7.1 - Cobertura total del código

El código, ha sido testeado a un 86% debido a que por falta de tiempo hay algunos casos que no se han podido testar. La mayoría de la falta de cobertura es por los casos incorrectos de cada test. Como objetivo principal, me puse el superar el 80% de cobertura del proyecto.

Las herramientas utilizadas para sacar el informe de cobertura es Jacoco.

GESTOR-ITV

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
dev.kkarrasmil80.gestoritv.vehiculo.storage		79 %		54 %	21	36	22	154	4	12	1	5
dev.kkarrasmil80.gestoritv		0 %		n/a	5	5	18	18	5	5	3	3
dev.kkarrasmil80.gestoritv.vehiculo.service		88 %		83 %	1	10	2	26	0	7	0	2
dev.kkarrasmil80.gestoritv.cita.service		87 %		80 %	2	11	2	23	0	6	0	1
dev.kkarrasmil80.gestoritv.cita.repositories		81 %		0 %	2	8	1	14	1	7	0	1
dev.kkarrasmil80.gestoritv.vehiculo.validator		100 %		100 %	0	49	0	90	0	8	0	4
dev.kkarrasmil80.gestoritv.vehiculo.repositories		100 %		100 %	0	8	0	13	0	7	0	2
dev.kkarrasmil80.gestoritv.vehiculo.dao		100 %		n/a	0	14	0	14	0	14	0	1
dev.kkarrasmil80.gestoritv.cliente.service		100 %		100 %	0	6	0	11	0	3	0	1
dev.kkarrasmil80.gestoritv.cliente.validator		100 %		100 %	0	6	0	10	0	2	0	1
dev.kkarrasmil80.gestoritv.cliente.repositories		100 %		100 %	0	5	0	4	0	3	0	1
dev.kkarrasmil80.gestoritv.cita.validator		100 %		100 %	0	5	0	8	0	2	0	1
dev.kkarrasmil80.gestoritv.cliente.dao		100 %		n/a	0	5	0	5	0	5	0	1
dev.kkarrasmil80.gestoritv.cita.dao		100 %		n/a	0	5	0	5	0	5	0	1
Total	306 of 2.289	86 %	26 of 172	84 %	31	173	45	395	10	86	4	25

Test Summary

128	0	0	1.753s	100%
tests	failures	ignored	duration	successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
cita.dao	8	0	0	1.276s	100%
cita.repositories	8	0	0	0.063s	100%
cita.service	9	0	0	0.079s	100%
cita.validator	4	0	0	0.008s	100%
cliente.dao	5	0	0	0.055s	100%
cliente.repositories	4	0	0	0.007s	100%
cliente.service	5	0	0	0.013s	100%
cliente.validator	5	0	0	0.003s	100%
vehiculo.dao	8	0	0	0.096s	100%
vehiculo.repositories	9	0	0	0.011s	100%
vehiculo.service	8	0	0	0.016s	100%
vehiculo.storage	10	0	0	0.107s	100%
vehiculo.validator	45	0	0	0.019s	100%

8. - ESTIMACIÓN DE COSTES Y PREPARACION

Aquí se muestran las horas estimadas y las horas reales invertidas a lo largo del ciclo de vida del proyecto

NÚMERO	Requisitos Funcionales	Horas estimadas	Horas real	Diferencia
1	Operaciones CRUD	5	10	5
2	Diferenciar Vehículos	1	2	1
3	Validador	1	1	0
4	Validar datos de introducción	2	5	3
5	Inicio de sesión sólido	3	4	1

6	Permitir importación	10	11	1
7	Generar informe	8	13	5
8	Vistas	2	3	1
9	Lógica de negocio	30	50	
TOTAL HORAS			99	

Aquí se muestran los gastos totales de todo el proyecto durante su duración

CONCEPTO	DETALLE	CÁLCULO	ESTIMACIÓN
RRHH	desarrollador	5 días a la semana * 6.5 horas diarias = 146.25 horas	146.25 horas * 35 euros la hora = 5.118,75 €
LICENCIAS Y HERRAMIENTAS	IDE (IntelliJ IDEA)	Licencia empresa	200 €
	Whismical premium	Licencia empresa	20 €
	LucidChart	Licencia empresa	20 €
	Draw.io	Licencia gratuita	0 €
FORMACIÓN	Tiempo en aprender nuevas librerías para la ejecución del proyecto	10 * 35 € hora = 350 €	350 €

Gastos varios	Electricidad, café, internet...		60 €
—	—	—	—
TOTAL DEL PROYECTO			5768,75 €

9. - CONCLUSIONES

El proyecto, ha sido difícil desempeñarlo pero con esfuerzo, dedicación y muchas horas de trabajo, se ha conseguido. Es algo de lo que estoy bastante orgulloso ya que es por así decirlo mi primer proyecto “grande” entre muchas comillas. En general este tipo de proyectos suponen un caos mental muy fuerte y un desorden de ideas muy claro, pero todo es ponerse y sacarlo en claro.