

Universidad de Castilla La Mancha
Escuela Superior de Informática



Procesos de Ingeniería Software 2023-2024

G1:

Yolanda Ayuso Agúndez

Yolanda.Ayuso@alu.uclm.es

Cristian Rubio Barato

Cristian.Rubio@alu.uclm.es

Alejandro Sánchez Arcos

Alejandro.Sanchez54@alu.uclm.es

Alejandro Medina Valderas

Alejandro.Medina6@alu.uclm.es

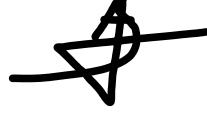
Julián Román Alberca

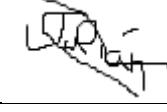
Julian.Roman@alu.uclm.es

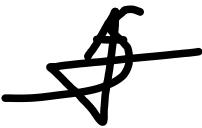
ÍNDICE

1	Plantilla de Participación (Sprint)	3
2	Análisis: Especificación de Requisitos y Descripción Funcional de la Aplicación	4
3	Diseño.....	32
3.1	Arquitectura Tecnológica.....	32
3.2	Procesos	33
3.2.1	Gestión de Código: Repositorio y Organización	33
3.2.2	Integración Continua	34
3.2.3	Despliegue.....	35
3.2.4	Pruebas.....	37
3.3	TDD	38
4	Desarrollo	39
4.1	Arquitectura del Proyecto	39
4.2	Seguridad y Gestión de Roles	43
4.3	Calidad.....	45
4.4	Limitaciones Encontradas.....	48
5	Manual de Usuario	49
6	Mantenimiento.....	73
6.1	Análisis Crítico del Proyecto Heredado	73
6.2	Alcance Definido.....	75
6.3	Addenda a la arquitectura de desarrollo.....	76

1 Plantilla de Participación (Sprint)

Sprint 1		
Apellidos y Nombre	Firma	Participación (%)
Yolanda Ayuso Agúndez		20
Cristian Rubio Barato		20
Julián Román Alberca		20
Alejandro Medina Valderas		20
Alejandro Sánchez Arcos		20

Sprint 2		
Apellidos y Nombre	Firma	Participación (%)
Yolanda Ayuso Agúndez		20
Cristian Rubio Barato		20
Julián Román Alberca		20
Alejandro Medina Valderas		20
Alejandro Sánchez Arcos		20

Sprint 3		
Apellidos y Nombre	Firma	Participación (%)
Yolanda Ayuso Agúndez		20
Cristian Rubio Barato		20
Julián Román Alberca		20
Alejandro Medina Valderas		20
Alejandro Sánchez Arcos		20

2 Análisis: Especificación de Requisitos y Descripción Funcional de la Aplicación

Hemos establecido una clasificación basándonos en los diferentes procesos involucrados en el ciclo de vida del software. Consideramos que, al dividir los requisitos en categorías basadas en los diferentes procesos, se puede entender mejor cómo cada requisito contribuye al funcionamiento general del software y cómo interactúa con los demás requisitos. Esto también puede ayudar a los equipos de desarrollo a priorizar y asignar recursos de manera más efectiva. A continuación, se muestran los diferentes procesos con los requisitos que los integran y toda la información de ellos.

2.1 PROCESO REGISTRO ROLES

SECCIÓN	CONTENIDO
ID	REQ01
NOMBRE	Registro usuarios (TDD)
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario podrá registrarse en la aplicación.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:**C.A.1.** Registro exitoso

- Los usuarios deben poder completar el proceso de registro con éxito proporcionando los campos "correo electrónico", "contraseña", "fecha de nacimiento", "DNI", "número de teléfono".
- El sistema debe proporcionar validación de correo electrónico para asegurar que se ingrese una dirección válida y única.

C.A.2. Validación de contraseña

- La contraseña debe cumplir con los requisitos de seguridad establecidos en el documento CCN-CERT BP/28, es decir, longitud mínima, uso de mayúsculas y minúsculas y números y caracteres especiales.
- El sistema proporcionará retroalimentación clara sobre si los requisitos de contraseña se cumplen.

C.A.3. Validación de campos obligatorios

- Todos los campos obligatorios "correo electrónico", "contraseña", "fecha de nacimiento", "DNI", "número de teléfono" deben de ser validados antes de permitir el registro.
- Los usuarios no deben poder registrarse sin completar los campos requeridos.

SECCIÓN	CONTENIDO
ID	REQ02
NOMBRE	Crear administrador
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador tendrá la capacidad de añadir nuevos administradores
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:**C.A.1.** Registro exitoso

- Los administradores podrán añadir satisfactoriamente a otros administradores proporcionando al sistema el nombre, apellidos, DNI, correo, ciudad, contraseña.
- El sistema debe proporcionar validación de correo electrónico para asegurar que se ingrese una dirección válida y única

C.A.2. Validación de contraseña

- La contraseña debe cumplir con los requisitos de seguridad establecidos en el documento CCN-CERT BP/28, es decir, longitud mínima, uso de mayúsculas y minúsculas y números y caracteres especiales.
- El sistema proporcionará retroalimentación clara sobre si los requisitos de contraseña se cumplen.

C.A.3. Validación de campos obligatorios

- Todos los campos obligatorios "correo electrónico", "contraseña", "nombre", "apellidos", "DNI" deben de ser validados antes de permitir el registro.
- Los administradores no deben poder añadirse sin completar los campos requeridos

SECCIÓN	CONTENIDO
ID	REQ03
NOMBRE	Crear miembro de personal de mantenimiento
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador tendrá la capacidad de añadir nuevos personal de mantenimiento
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

C.A.1. Registro exitoso

- Los administradores podrán añadir satisfactoriamente a personal de mantenimiento proporcionando al sistema el nombre, apellidos, DNI, correo, ciudad, contraseña, carnet y experiencia en años en la empresa.
- El sistema debe proporcionar validación de correo electrónico para asegurar que se ingrese una dirección válida y única

C.A.2. Validación de contraseña

- La contraseña debe cumplir con los requisitos de seguridad establecidos en el documento CCN-CERT BP/28, es decir, longitud mínima, uso de mayúsculas y minúsculas y números y caracteres especiales.
- El sistema proporcionará retroalimentación clara sobre si los requisitos de contraseña se cumplen.

C.A.3. Validación de campos obligatorios

- Todos los campos obligatorios nombre, apellidos, DNI, correo, experiencia y contraseña deben de ser validados antes de permitir el registro.
- Los administradores no deben poder añadir personal de mantenimiento sin completar los campos requeridos

2.2 PROCESO INICIO DE SESIÓN

SECCIÓN	CONTENIDO
ID	REQ04
NOMBRE	Iniciar Sesión usuarios
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe tener la capacidad de acceder al sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Inicio de sesión satisfactorio con credenciales existentes
 - El usuario debe poder ingresar su nombre de usuario y contraseña válidos utilizando el patrón RBAC.
 - Si los datos son correctos, el sistema permitirá al usuario acceder a su cuenta.
 - El usuario será redirigido a la página principal después del inicio de sesión.
- **C.A.2.** Manejo de datos incorrectos.
 - Si el usuario ingresa su nombre de usuario y contraseña incorrecto, debe recibir un mensaje de error adecuado.
 - El mensaje de error debe indicar que los datos ingresados son incorrectos sin revelar información sobre la existencia de la cuenta.
- **C.A.3.** Restablecimiento de contraseña.
 - Los usuarios deben tener la opción de restablecer su contraseña en caso de olvidarla.
 - El sistema proporcionará un proceso intuitivo para solicitar restablecer la contraseña a través de un correo electrónico.
- **C.A.4.** Inicio de sesión seguro.
 - Para iniciar sesión el usuario deberá realizar un doble factor de autenticación.
 - El inicio de sesión debe de cumplir con los principios de seguridad establecidos en el documento CCN-CERT BP/28.
 - El sistema debe desactivar al usuario tras 5 intentos de inicio de sesión no válidos.

SECCIÓN	CONTENIDO
ID	REQ05
NOMBRE	Iniciar Sesión personal de mantenimiento
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol personal de mantenimiento debe tener la capacidad de acceder al sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Inicio de sesión satisfactorio con credenciales existentes
 - El personal de mantenimiento debe poder ingresar su nombre de usuario y contraseña válidos utilizando el patrón RBAC.
 - Si los datos son correctos, el sistema permitirá al personal de mantenimiento acceder a su cuenta.
 - El personal de mantenimiento será redirigido a la página principal después del inicio de sesión.
- **C.A.2.** Manejo de datos incorrectos.
 - Si el personal de mantenimiento ingresa su nombre de usuario y contraseña incorrecto, debe recibir un mensaje de error adecuado.
 - El mensaje de error debe indicar que los datos ingresados son incorrectos sin revelar información sobre la existencia de la cuenta.
- **C.A.3.** Inicio de sesión seguro.
 - Para iniciar sesión el personal de mantenimiento deberá realizar un doble factor de autenticación.
 - El inicio de sesión debe de cumplir con los principios de seguridad establecidos en el documento CCN-CERT BP/28.
 - El sistema debe desactivar al personal de mantenimiento tras 5 intentos de inicio de sesión no válidos.

SECCIÓN	CONTENIDO
ID	REQ06
NOMBRE	Iniciar Sesión administradores
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la capacidad de acceder al sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Inicio de sesión satisfactorio con credenciales existentes
 - El administrador debe poder ingresar su nombre de usuario y contraseña válidos utilizando el patrón RBAC.
 - Si los datos son correctos, el sistema permitirá al administrador acceder a su cuenta.
 - El administrador será redirigido a la página principal después del inicio de sesión.
- **C.A.2.** Manejo de datos incorrectos.
 - Si el administrador ingresa su nombre de usuario y contraseña incorrecto, debe recibir un mensaje de error adecuado.
 - El mensaje de error debe indicar que los datos ingresados son incorrectos sin revelar información sobre la existencia de la cuenta.
- **C.A.3.** Inicio de sesión seguro.
 - Para iniciar sesión el administrador deberá realizar un doble factor de autenticación.
 - El inicio de sesión debe de cumplir con los principios de seguridad establecidos en el documento CCN-CERT BP/28.
 - El sistema debe desactivar al administrador tras 5 intentos de inicio de sesión no válidos.

2.3 PROCESO DESACTIVAR Y ELIMINAR CUENTAS

SECCIÓN	CONTENIDO
ID	REQ07
NOMBRE	Desactivar administradores
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder desactivar la cuenta de otros administradores
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar otros administradores registrados en el sistema.
 - Validación previa a la eliminación de un administrador para mitigar la posibilidad de eliminación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ08
NOMBRE	Desactivar personal de mantenimiento
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder desactivar la cuenta de personal de mantenimiento
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar personal de mantenimiento en el sistema.
 - Validación previa a la eliminación de un personal de mantenimiento para mitigar la posibilidad de eliminación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ09
NOMBRE	Desactivar usuarios
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder desactivar la cuenta de usuarios registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar usuarios registrados en el sistema.
 - Validación previa a la eliminación de un usuario para mitigar la posibilidad de eliminación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ10
NOMBRE	Desactivar usuarios
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario tendrá la facultad de poder eliminar su cuenta de forma definitiva del sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de usuario
 - La opción para eliminar la cuenta debe estar disponible en la interfaz de usuario de la sección de configuración de la cuenta del usuario.
 - La opción de eliminar la cuenta debe ser clara y fácil de encontrar.
- **C.A.2.** Confirmación
 - Antes de proceder con la eliminación, el sistema debe requerir una confirmación explícita por parte del usuario.
 - La confirmación debe incluir un mensaje claro que indique las consecuencias de la acción.
- **C.A.3.** Seguridad
 - El proceso de eliminación de la cuenta debe garantizar la autenticación del usuario para prevenir eliminaciones no autorizadas.

2.4 PROCESO CONSULTAR CUENTAS

SECCIÓN	CONTENIDO
ID	REQ11
NOMBRE	Consultar administradores
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder consultar los administradores registrados en el sistema, así como su información de perfil.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de consulta de administradores registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de administradores
 - La consulta debe proporcionar un listado completo de todos los administradores registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un administrador específico del listado, se debe mostrar su información de perfil de manera detallada.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de administradores.
 - No se podrá modificar ningún campo en la consulta.

SECCIÓN	CONTENIDO
ID	REQ12
NOMBRE	Consultar personal de mantenimiento
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder consultar los personales de mantenimiento registrados en el sistema, así como su información de perfil.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de consulta de personales de mantenimiento registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de personales de mantenimiento
 - La consulta debe proporcionar un listado completo de todos los personales de mantenimiento registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un personal de mantenimiento específico del listado, se debe mostrar su información de perfil de manera detallada.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de personales de mantenimiento.
 - No se podrá modificar ningún campo en la consulta.

SECCIÓN	CONTENIDO
ID	REQ13
NOMBRE	Consultar usuarios
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder consultar los usuarios registrados en el sistema, así como su información de perfil.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de consulta de usuarios registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de usuarios
 - La consulta debe proporcionar un listado completo de todos los usuarios registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un usuario específico del listado, se debe mostrar su información de perfil de manera detallada.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de usuarios.
 - No se podrá modificar ningún campo en la consulta.

SECCIÓN	CONTENIDO
ID	REQ14
NOMBRE	Consultar mi información de perfil
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario tendrá la capacidad de consultar su información de perfil
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de usuario debe tener acceso a la función de consulta de perfil.
 - La función debe ser fácilmente accesible desde la interfaz de usuario.
- **C.A.2.** Información de perfil

- Al seleccionar la opción, se debe mostrar su información de perfil de manera detallada.
- **C.A.4.** Seguridad y privacidad
 - No se podrá modificar ningún campo en la consulta.

2.5 PROCESO MODIFICAR CUENTAS

SECCIÓN	CONTENIDO
ID	REQ15
NOMBRE	Modificar administradores
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder modificar la información de perfil de los administradores registrados en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de modificación de administradores registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de administradores
 - La consulta debe proporcionar un listado completo de todos los administradores registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un administrador específico del listado, se debe mostrar su información de perfil de manera detallada y poder modificar los campos permitidos.
 - Se podrá modificar los campos: nombre, apellidos, DNI, ciudad.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de administradores.
 - El campo correo electrónico no podrá modificarse.

SECCIÓN	CONTENIDO
ID	REQ16
NOMBRE	Modificar personal de mantenimiento
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder modificar la información de perfil de personales de mantenimiento registrados en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de modificación de personales de mantenimiento registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de personales de mantenimiento
 - La consulta debe proporcionar un listado completo de todos los personales de mantenimiento registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un personal de mantenimiento específico del listado, se debe mostrar su información de perfil de manera detallada y poder modificar los campos permitidos.
 - Se podrá modificar los campos: nombre, apellidos, DNI, ciudad, carnet, experiencia.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de administradores.
 - El campo correo electrónico no podrá modificarse.

SECCIÓN	CONTENIDO
ID	REQ17
NOMBRE	Modificar usuarios
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder modificar la información de perfil de usuarios registrados en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de administrador debe tener acceso a la función de modificación de usuarios registrados.
 - La función debe ser fácilmente accesible desde la interfaz de administración.
- **C.A.2.** Listado de usuarios
 - La consulta debe proporcionar un listado completo de todos los usuarios registrados en el sistema.
- **C.A.3.** Información de perfil
 - Al seleccionar un usuario específico del listado, se debe mostrar su información de perfil de manera detallada y poder modificar los campos permitidos.
 - Se podrá modificar los campos: nombre, apellidos, DNI, teléfono, carnet, fecha de nacimiento.
- **C.A.4.** Seguridad y privacidad
 - La información sensible, como contraseñas, no debe ser visible en la consulta de administradores.
 - El campo correo electrónico no podrá modificarse.

SECCIÓN	CONTENIDO
ID	REQ18
NOMBRE	Modificar mi perfil
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario tendrá la capacidad de modificar su información de perfil en su cuenta una vez logueado
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Acceso a la funcionalidad
 - El rol de usuario debe tener acceso a la función de modificación de perfil.
 - La función debe ser fácilmente accesible desde la interfaz de usuario.
- **C.A.2.** Información de perfil
 - Se podrá modificar los campos: nombre, apellidos, DNI, teléfono, carnet, fecha de nacimiento y contraseña.
- **C.A.3.** Seguridad y privacidad
 - El campo correo electrónico no podrá modificarse.
- **C.A.4.** Validación de campos.
 - La contraseña debe cumplir con los requisitos de seguridad establecidos en el documento CCN-CERT BP/28, es decir, longitud mínima, uso de mayúsculas y minúsculas y números y caracteres especiales.

- El sistema proporcionará retroalimentación clara sobre si los requisitos de contraseña se cumplen.

2.6 PROCESO AÑADIR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ19
NOMBRE	Añadir coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para añadir coches en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para agregar coches.
 - La interfaz debe incluir campos para ingresar información relevante: tipo, matrícula, número de plazas, dirección, modelo, porcentaje de batería y estado.
- **C.A.2.** Validación de Datos:
 - El sistema debe realizar validaciones para asegurarse de que los datos ingresados para el nuevo coche sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores autorizados deben tener la capacidad de agregar coches al sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la adición de un nuevo coche.
- **C.A.4.** Campos Obligatorios:
 - Deben introducirse obligatoriamente para tener éxito en la operación los siguientes campos: tipo, matrícula, número de plazas, dirección, modelo, porcentaje de batería y estado.

SECCIÓN	CONTENIDO
ID	REQ20
NOMBRE	Añadir motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para añadir motos en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para agregar motos.
 - La interfaz debe incluir campos para ingresar información relevante: tipo, matrícula, casco, dirección, modelo, porcentaje de batería y estado.
- **C.A.2.** Validación de Datos:
 - El sistema debe realizar validaciones para asegurarse de que los datos ingresados para la nueva moto sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores autorizados deben tener la capacidad de agregar motos al sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la adición de una nueva moto.
- **C.A.4.** Campos Obligatorios:
 - Deben introducirse obligatoriamente para tener éxito en la operación los siguientes campos: tipo, matrícula, dirección, casco, modelo, porcentaje de batería y estado.

SECCIÓN	CONTENIDO
ID	REQ21
NOMBRE	Añadir patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para añadir patinetes en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para agregar patinetes.
 - La interfaz debe incluir campos para ingresar información relevante: tipo, matrícula, color, dirección, modelo, porcentaje de batería y estado.
- **C.A.2.** Validación de Datos:
 - El sistema debe realizar validaciones para asegurarse de que los datos ingresados para el nuevo patinete sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores autorizados deben tener la capacidad de agregar patinetes al sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la adición de un nuevo patinete.

- **C.A.4.** Campos Obligatorios:
 - Deben introducirse obligatoriamente para tener éxito en la operación los siguientes campos: tipo, matrícula, dirección, modelo, porcentaje de batería y estado.
 - El campo color será opcional.

2.7 PROCESO CONSULTAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ22
NOMBRE	Consultar coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para consultar coches registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para consultar los coches registrados en el sistema.
- **C.A.2.** Seguridad:
 - Solo los administradores deben tener la capacidad de consultar los coches registrados en el sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la consulta de coches.
 - No debe poderse modificar ningún campo.

SECCIÓN	CONTENIDO
ID	REQ23
NOMBRE	Consultar motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para consultar motos registradas en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para consultar las motos registradas en el sistema.
- **C.A.2.** Seguridad:

- Solo los administradores deben tener la capacidad de consultar las motos registradas en el sistema.
- Se debe verificar la autenticación del administrador antes de permitir la consulta de motos.
- No debe poderse modificar ningún campo.

SECCIÓN	CONTENIDO
ID	REQ24
NOMBRE	Consultar patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para consultar patinetes registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para consultar los patinetes registrados en el sistema.
- **C.A.2.** Seguridad:
 - Solo los administradores deben tener la capacidad de consultar los patinetes registrados en el sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la consulta de patinetes.
 - No debe poderse modificar ningún campo.

2.8 PROCESO MODIFICAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ25
NOMBRE	Modificar coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para modificar coches registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para modificar coches.
- **C.A.2.** Validación de Datos:

- El sistema debe realizar validaciones para asegurarse de que los datos ingresados para la modificación del coche sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores deben tener la capacidad de modificar coches al sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la modificación del coche.
- **C.A.4.** Campos Obligatorios:
 - Se podrán modificar todos los campos excepto matrícula.

SECCIÓN	CONTENIDO
ID	REQ26
NOMBRE	Modificar motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para modificar motos registradas en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para modificar motos.
- **C.A.2.** Validación de Datos:
 - El sistema debe realizar validaciones para asegurarse de que los datos ingresados para la modificación de la moto sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores deben tener la capacidad de modificar motos en el sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la modificación de la moto.
- **C.A.4.** Campos Obligatorios:
 - Se podrán modificar todos los campos excepto matrícula.

SECCIÓN	CONTENIDO
ID	REQ27
NOMBRE	Modificar patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para modificar patinetes registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

- **C.A.1.** Interfaz de Usuario:
 - Se debe proporcionar una interfaz clara y accesible en el panel de administración para modificar patinetes.
- **C.A.2.** Validación de Datos:
 - El sistema debe realizar validaciones para asegurarse de que los datos ingresados para la modificación del patinete sean correctos y cumplan con los requisitos establecidos (matrícula con 4 números y 3 letras).
- **C.A.3.** Seguridad:
 - Solo los administradores deben tener la capacidad de modificar patinetes al sistema.
 - Se debe verificar la autenticación del administrador antes de permitir la modificación del patinete.
- **C.A.4.** Campos Obligatorios:
 - Se podrán modificar todos los campos excepto matricula.

2.9 PROCESO DESACTIVAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ28
NOMBRE	Desactivar coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para desactivar coches registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar coches registrados en el sistema.
 - Validación previa a la desactivación de un coche para mitigar la posibilidad de desactivación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ29
NOMBRE	Desactivar motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para desactivar motos registradas en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar motos registrados en el sistema.
 - Validación previa a la desactivación de una moto para mitigar la posibilidad de desactivación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ30
NOMBRE	Desactivar patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para desactivar patinetes registrados en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán desactivar patinetes registrados en el sistema.
 - Validación previa a la desactivación de un patinete para mitigar la posibilidad de desactivación accidental mediante una confirmación adicional antes de proceder con la acción.

2.10 PROCESO ACTIVAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ31
NOMBRE	Activar coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para activar coches registrados en el sistema que estén desactivados.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán activar coches registrados en el sistema.
 - Validación previa a la activación de un coche para mitigar la posibilidad de activación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ32
NOMBRE	Activar motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para activar motos que estén desactivadas en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán activar motos registrados en el sistema.
 - Validación previa a la activación de una moto para mitigar la posibilidad de activación accidental mediante una confirmación adicional antes de proceder con la acción.

SECCIÓN	CONTENIDO
ID	REQ33
NOMBRE	Activar patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la facultad para activar patinetes registrados en el sistema que estén desactivados.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Los administradores del sistema podrán activar patinetes registrados en el sistema.
 - Validación previa a la activación de un patinete para mitigar la posibilidad de activación accidental mediante una confirmación adicional antes de proceder con la acción.

2.11 PROCESO RESERVAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ34
NOMBRE	Reservar coches
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder tener la facultad de reservar coches en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios realicen reservas de coches.
 - La interfaz debe proporcionar información relevante sobre los coches disponibles.
- **C.A.2.** Verificación de permisos.
 - El sistema deberá verificar que el usuario tiene carnet B para poder reservar coches.
 - El sistema debe verificar que el usuario no tiene ninguna reserva activa.

SECCIÓN	CONTENIDO
ID	REQ35
NOMBRE	Reservar motos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder tener la facultad de reservar motos en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios realicen reservas de motos.
 - La interfaz debe proporcionar información relevante sobre las motos disponibles.
- **C.A.2.** Verificación de permisos.
 - El sistema deberá verificar que el usuario tiene carnet A para poder reservar motos.
 - El sistema debe verificar que el usuario no tiene ninguna reserva activa.

SECCIÓN	CONTENIDO
ID	REQ36
NOMBRE	Reservar patinetes
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder tener la facultad de reservar patinetes en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios realicen reservas de patinetes.
 - La interfaz debe proporcionar información relevante sobre los patinetes disponibles.
- **C.A.2.** Verificación de permisos.
 - El sistema debe verificar que el usuario no tiene ninguna reserva activa.

SECCIÓN	CONTENIDO
ID	REQ37
NOMBRE	Valorar servicio de reserva
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder valorar el servicio de reserva una vez finalizada esta.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios valoren el servicio tras finalizar la reserva.
- **C.A.2.** Campos obligatorios y entradas de información:
 - El sistema debe verificar que se ha introducido obligatoriamente una puntuación del 1 al 5.
 - El sistema debe finalizar satisfactoriamente la operación, aunque el comentario no se haya introducido, ya que es opcional.
 - El sistema debe dar feedback al usuario de que no ha introducido una puntuación
 - El sistema debe restringir el número de caracteres del comentario a 50.
 - El sistema debe restringir el número de puntuación entre 1 y 5.

SECCIÓN	CONTENIDO
ID	REQ39
NOMBRE	Finalizar reserva
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder finalizar su reserva activa
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios finalicen la reserva que tienen activa.
 - Antes de que la reserva se complete de manera definitiva, el sistema debe mostrar una alerta de confirmación.
 - La alerta debe proporcionar información clara sobre la acción que está a punto de realizar el usuario, solicitando su confirmación. Debería incluir un mensaje que indique las consecuencias de finalizar la reserva y ofrecer opciones adicionales, como "Confirmar" o "Cancelar". Esta alerta tiene como objetivo evitar acciones accidentales y garantizar que el usuario realmente desea completar la reserva
- **C.A.2.** Operaciones:
 - Tras finalizar la reserva, el vehículo debe actualizar su porcentaje de batería.
 - Tras finalizar la reserva el vehículo debe cambiar su estado.
 - Tras finalizar la reserva el sistema debe activar el servicio de valoración para dicha reserva.
 - Tras finalizarla se debe efectuar el proceso de facturación.

SECCIÓN	CONTENIDO
ID	REQ40
NOMBRE	Cancelar reserva
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder cancelar su reserva activa
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios cancelen la reserva que tienen activa.

- Antes de que la reserva se complete de manera definitiva, el sistema debe mostrar una alerta de confirmación.
- La alerta debe proporcionar información clara sobre la acción que está a punto de realizar el usuario, solicitando su confirmación. Debería incluir un mensaje que indique las consecuencias de cancelar la reserva y ofrecer opciones adicionales, como "Confirmar" o "Cancelar". Esta alerta tiene como objetivo evitar acciones accidentales y garantizar que el usuario realmente desea cancelar la reserva
- **C.A.2. Operaciones:**
 - Tras cancelar la reserva el vehículo debe cambiar su estado.

SECCIÓN	CONTENIDO
ID	REQ41
NOMBRE	Consultar reserva
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder consultar la información de su reserva activa
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1. Interfaz de Usuario:**
 - Debe existir una interfaz de usuario clara y fácil de usar para que los usuarios visualicen la reserva que tienen activa.

SECCIÓN	CONTENIDO
ID	REQ42
NOMBRE	Consultar reservas del sistema
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe poder consultar el listado de reservas realizadas en el sistema
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1. Interfaz de Usuario:**
 - Debe existir una interfaz de administrador clara y fácil de usar para que los administradores visualicen el listado de reservas realizadas en el sistema.
 - Debe proporcionar el usuario, la matrícula, modelo y fecha de la reserva para identificarla.

2.12 PROCESO PERSONAL DE MANTENIMIENTO RECARGAR VEHÍCULOS

SECCIÓN	CONTENIDO
ID	REQ43
NOMBRE	Reservar vehículos para recargar
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol personal de mantenimiento tendrá la facultad de reservar vehículos para cargarlos
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el personal de mantenimiento realicen reservas de vehículos para recargar.
 - La interfaz debe proporcionar información relevante sobre los vehículos en estado de recarga.
- **C.A.2.** Verificación de permisos.
 - El sistema debe validar que el personal de mantenimiento tiene permiso para recargar vehículos, ya que únicamente puede recargar un máximo de 5 vehículos a la vez.

SECCIÓN	CONTENIDO
ID	REQ44
NOMBRE	Cargar vehículos
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol personal de mantenimiento tendrá la facultad de cargar los vehículos.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el personal de mantenimiento cargue el vehículo.
- **C.A.2.** Operaciones:
 - Tras cargar el vehículo, este modificará su porcentaje de batería al 100%.
 - Tras cargar el vehículo, este cambiará su estado a disponible.

2.13 PROCESO AÑADIR TIPO DE VEHÍCULO

SECCIÓN	CONTENIDO
ID	REQ45
NOMBRE	Añadir tipo de vehículo en el sistema
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la capacidad de añadir un nuevo tipo de vehículo en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el administrador añada un nuevo tipo de vehículo, proporcionando el nombre de este.

SECCIÓN	CONTENIDO
ID	REQ46
NOMBRE	Eliminar tipo de vehículo en el sistema
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador debe tener la capacidad de eliminar un tipo de vehículo en el sistema.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el administrador elimine un tipo de vehículo registrado en el sistema.

2.14 REQUISITOS PREMIUM

2.14.1 PROCESO VISUALIZAR HISTORIAL DE RESERVAS

SECCIÓN	CONTENIDO
ID	REQ47
NOMBRE	Visualizar historial de reservas
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe tener la facultad de visualizar un listado con sus reservas, tanto históricas como canceladas.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el usuario visualice un listado de reservas tanto históricas como canceladas.
 - Cada reserva debe proporcionar la fecha, modelo, matrícula y estado.

2.14.2. PROCESO VISUALIZAR VALORACIONES

SECCIÓN	CONTENIDO
ID	REQ47
NOMBRE	Visualizar valoraciones para un vehículo
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol administrador podrá visualizar un listado con el detalle de todas las valoraciones para un determinado vehículo.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el usuario visualice un listado de vehículos y elija uno para obtener todas sus valoraciones.
 - Al elegir el vehículo deseado, se debe mostrar al usuario las valoraciones realizadas para ese vehículo. Proporcionando usuario que la realizó, comentario y puntuación.

2.14.3. PROCESO VISUALIZAR MEDIA DE PUNTUACIÓN

SECCIÓN	CONTENIDO
ID	REQ47
NOMBRE	Visualizar media de puntuación para un vehículo
VERSIÓN	V1.0
TIPO	R. F
DESCRIPCIÓN	El rol usuario debe poder visualizar la puntuación media de cada vehículo del sistema disponible para reservar.
CRITICIDAD	high
ESTADO	Proposed
ESFUERZO	50 sp

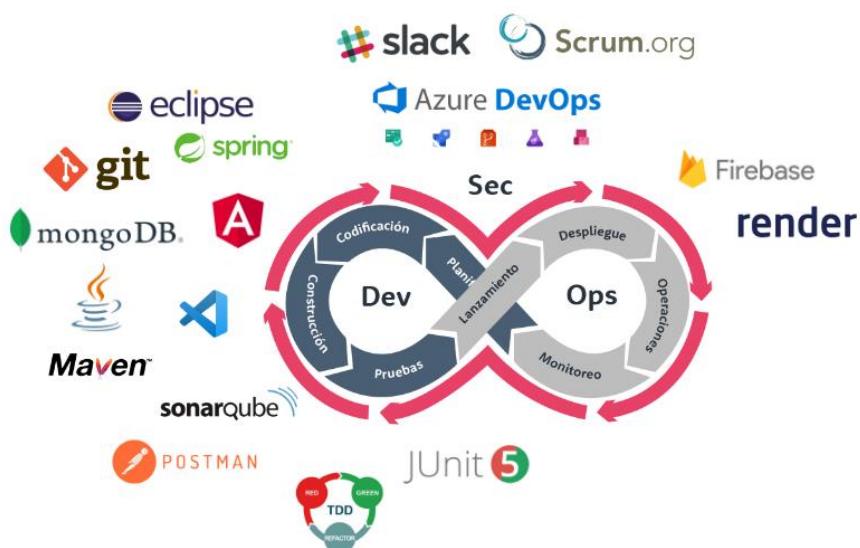
CRITERIOS DE ACEPTACIÓN:

- **C.A.1.** Interfaz de Usuario:
 - Debe existir una interfaz de usuario clara y fácil de usar para que el usuario visualice la media de la puntuación de cada vehículo al lado de su matrícula y modelo.

3 Diseño

3.1 Arquitectura Tecnológica

La Arquitectura Tecnológica usada en el proyecto integrado se resume en la siguiente imagen del ciclo DevSecOps:



En la parte de **Desarrollo** del Ciclo DevSecOps, para la parte de Planificación se ha utilizado como marco de trabajo Scrum con Slack para la comunicación y gestión de incidencias del equipo y Azure Boards para la gestión y planificación del proyecto.

Para la parte de codificación los IDE's usados han sido Eclipse para el desarrollo de la parte Backend con Spring y Java, y para la parte Frontend Visual Studio Code con Angular. La gestión del código y de versiones se ha utilizado git en Azure Repos. La base de datos se ha implantado en Mongo DB.

Para la parte de compilación y pruebas se ha utilizado Maven, Postman, Sonarqube para realizar un seguimiento de la calidad y TDD con Junit 5 para la realización de las pruebas.

En la parte de **Operaciones** del ciclo DevSecOps para lograr la integración continua y el despliegue se ha utilizado Azure Pipelines y las plataformas para el despliegue han sido Firebase (Frontend) y Render(Backend).

Para completar el ciclo DevSecOps, hay que destacar que la **Seguridad** se ha tenido en cuenta durante todas las fases del ciclo.

3.2 Procesos

3.2.1 Gestión de Código: Repositorio y Organización

El objetivo de la gestión de código en MueveTIC es asegurar que todos los elementos que conforman el producto que se ha elaborado, sean gestionados de manera sistemática y controlada desde su concepción hasta su implementación y mantenimiento.

A continuación, se detalla los aspectos clave de la gestión realizada:

3.2.1.1. Repositorios

Se ha usado repositorios git en Azure. Se ha utilizado un repositorio para Frontend y otro Backend.

- Branching del repositorio:

Branch	Co...	Author	Authored...
develop-AlejandroSan	9e50361	AlejandroSan...	29 nov
develop-general	cd722b0	JULIÁN ROM...	30 nov
develop-Yolanda	888fad1	YOLANDA AY...	30 nov
master	9305ea81	JULIÁN ROM...	30 nov

Branch	Co...	Author	Authored...
develop-AlexM	b64a951	MedinaMV	30 nov
develop-Cristian	3e26f51	JULIÁN ROM...	28 nov
Develop-General	05a28e1	CRISTIAN RU...	21 nov
develop-Julian	0563251	JulianRA	29 nov
develop-Yolanda	a9c1401	YOLANDA AY...	27 nov
master	c71a111	JULIÁN ROM...	30 nov
QualitySonar	30848a1	JulianRA	29 nov
test-AlexM	c3832f1	MedinaMV	28 nov
TestBranch	2f16131	CRISTIAN RU...	27 nov

En nuestro caso, usamos la rama principal (master), la develop-general, la develop específica de cada desarrollador, una para el testing y otra para analizar la calidad con SonarQube.

- Rama develop-<nombre>: Cada miembro del equipo de desarrollo tendrá una rama en la que desarrollara las funcionalidades que se ha asignado, además de realizar los cambios y correcciones en ella.
- Rama-general: El repositorio frontend y backend tendrán dicha rama donde se van mergeando las distintas funcionalidades provenientes de las ramas de cada desarrollador.
- Rama-master: En ella queda reflejado el código que se presenta como incremento del sprint.

- QualitySonar: rama en la que se corrigen aspectos de calidad reportados por Sonarqube.
- TestBranch: rama que contiene los test unitarios de la aplicación.

3.2.1.2. Control de cambios

En nuestro proyecto se realiza el siguiente procedimiento de control de cambios:

- Solicitud de Cambio: Cualquier miembro del equipo Scrum puede proponer un cambio.
- Revisión de Cambios: El equipo de desarrollo junto al product owner revisa la solicitud, evalúa su viabilidad y determina el impacto potencial en el producto.
- Aprobación: El Product Owner y, en algunos casos, el equipo de desarrollo aprueba o rechazan la solicitud en función de los criterios de aceptación y las necesidades del proyecto.
- Implementación: Una vez aprobado, el cambio se implementa en la configuración correspondiente, y se actualiza el control de versiones.

3.2.1.3. Gestión de versiones

En el marco de MueveTIC, se sigue una estrategia de versionado semántico para la gestión efectiva del ciclo de vida del software. Esta estrategia utiliza un formato "MAJOR.MINOR.PATCH" que tiene significados específicos en términos de cambios en el software.

- Desarrollo de nuevo incremento: Se incrementa el número MINOR para reflejar la adición de funcionalidades sin alterar la compatibilidad con el incremento del sprint anterior.
- Corrección de Errores: Se incrementa el número PATCH para indicar correcciones de errores sin afectar la compatibilidad con el incremento que tenemos del sprint anterior más el del sprint actual.
- Cambio Estructural Importante: Se incrementa el número MAJOR para señalar cambios estructurales importantes que pueden afectar la compatibilidad con el incremento.

3.2.2 Integración Continua

La realización de la integración continua la hemos realizado a través de "Azure Pipelines", en donde hemos creado un pipeline en la cual hemos definido distintos stages o etapas. En ellas realizamos las operaciones necesarias para el correcto funcionamiento como puede ser la etapa Build, la cual se centra en compilar el código fuente, gestionar dependencias y ejecutar pruebas o una etapa definida como "Sonarqube" donde ejecutaremos la herramienta sonar scanner para poder comprobar la calidad y requisitos especificados por el stakeholder correspondiente., lo cual nos generara un reporte de calidad del proyecto en el momento que se ejecute un cambio en el repositorio sincronizado.

```

trigger:
- develop-AlexM

pool:
  - name: Default
  - demands:
    - agent.name == CALSOFT2324

stages:
- stage: Preparation
  jobs:
    - job: PreparationJob
      steps:
        - script: echo Preparation!

- stage: Build
  jobs:
    - job: BuildJob
      steps:
        - script:
          - echo 'Running Maven Version'
          - mvn -version

      Settings
      task: Maven@3
      inputs:
        mavenPomFile: 'MueveTIC/pom.xml'
        mavenOptions: '-Xmx3072m'
        javaHomeOption: 'JDKVersion'
        jdkVersionOption: '1.17'
        jdkArchitectureOption: 'x64'
        publishJUnitResults: true
        testResultsFiles: '**/surefire-reports/TEST-*.xml'
        goals: 'package'

- stage: Sonarqube
  jobs:
    - job: SonarqubeJob
      steps:
        - checkout: none
        - script: sonar-scanner -Dsonar.token=$(SQ_TOKEN)
        - script: echo Sonarqube ok!

```

3.2.3 Despliegue

Frontend

Para realizar el despliegue de la parte Frontend se ha utilizado la plataforma de Firebase. Para ello hemos integrado el despliegue en la pipeline:

```
1 # Node.js with Angular
2 # Build a Node.js project that uses Angular.
3 # Add steps that analyze code, save build artifacts, deploy, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/languages/javascript
5
6
7 trigger:
8 - develop-Yolanda
9 -
10 pool:
11   - vmImage: ubuntu-latest
12   - name: Default
13   - demands:
14     - agent.name -equals AgenteFrontendYolanda
15   -
16 stages:
17 - stage: Preparation
18   - jobs:
19     - job: PreparationJob
20       - steps:
21         - script: echo Preparation!
22     -
23 - stage: Build
24   - jobs:
25     - job: BuildJob
26       - steps:
27         - checkout: self
28           Settings
29             - task: NodeTool@0
30               inputs:
31                 - versionSpec: '18.x'
32                 - displayName: 'Install Node.js'
33               - script: |
34                 npm install
35                 npm install -g @angular/cli
36                 npm install -g firebase-tools
37                 ng build --prod
38                 firebase deploy --token ${FIREBASE_TOKEN}
39                 - displayName: 'npm install and build'
40   -
41 - stage: Sonarqube
42   - jobs:
43     - job: SonarqubeJob
44       - steps:
45         - checkout: none
46         - script: sonar-scanner.bat -Dsonar.projectKey=ProyectoIntegrado_Frontend" -Dsonar.sources=". -Dsonar.host.url=http://localhost:9000" -Dsonar.token=${SO_TOKEN}"
47         - script: echo Sonarqube ok!
```

Como podemos observar en la pipeline, la línea para desplegar el código es la siguiente:

```
37 .....firebase deploy --token ${FIREBASE_TOKEN}
```

La variable `$(FIREBASE_TOKEN)` corresponde con el token de autorización que obtienes en tu cuenta al solicitarlo tras hacer el login en firebase para desplegarlo en un proyecto de tu cuenta.

Backend

Para el despliegue del backend hemos utilizado la plataforma de render. Para ello nos damos de alta y le damos a desplegar un nuevo servicio web. Para que todo esto funcione, debemos desplegarlo en Docker. Para ello, en la raíz del proyecto Spring debemos crear un archivo Dockerfile, aquí está el que hemos usado:

```
1 FROM maven:3.9.0-eclipse-temurin-17-alpine as build
2 COPY . .
3 RUN mvn clean package -Pprod -DskipTests
4
5 FROM eclipse-temurin:17-jdk-alpine
6 COPY --from=build target/*.jar app.jar
7 ENTRYPOINT ["java","-jar","/app.jar"]
8 EXPOSE 8080
```

Al vincular Github con Render, podemos desplegar el backend desde ahí cada vez que hagamos un commit a la rama main del proyecto (o la que nosotros elijamos en la configuración de Render)

3.2.4 Pruebas

Para este proceso, hemos incluido pruebas automatizadas para distintas historias de usuario acordadas con el Product Owner, por ejemplo, para el registro de un nuevo cliente a través de su DNI, número de teléfono o para poder comprobar si un usuario tiene una reserva activa lo que ocasionaría un conflicto o, por lo contrario, devolvería un código 200. Estas pruebas abarcan casos de éxito, donde se comprueba que el proceso de registro se complete correctamente bajo condiciones normales, así como casos de error, donde se evalúa la capacidad del sistema para gestionar situaciones inesperadas o datos incorrectos.

A continuación, se muestran algunas de las pruebas unitarias desarrolladas:

```
@Test @Order(1)
void testBookingCar() throws Exception{
    ResultActions OkeyData = this.sendCreateBooking("1234SAD", "Alejandro.Sanchez@alu.uclm.es");
    OkeyData.andExpect(status().isOk());
}

@Test @Order(2)
void testUserWithBookingPreviouslyCreated() throws Exception{
    ResultActions WrongData = this.sendCreateBooking("1234SAD", "Alejandro.Sanchez@alu.uclm.es");
    WrongData.andExpect(status().isConflict());
}

@Test @Order(3)
void testWrongCarnet() throws Exception{
    ResultActions WrongData = this.sendCreateBooking("1234SAD", "Yolanda.Ayuso@alu.uclm.es");
    WrongData.andExpect(status().isConflict());
}

@Test @Order(4)
void testBookingScooter() throws Exception{
    ResultActions OkeyData = this.sendCreateBooking("8975JJM", "Yolanda.Ayuso@alu.uclm.es");
    OkeyData.andExpect(status().isOk());
}
```

Pruebas unitarias Reservar Vehículos

```
@Test
@Order(1)
void chargeVehicle() throws Exception {
    this.addLowBatteryVehicle();
    RequestBuilder requestChargeVehicle = MockMvcRequestBuilders
        .get("/vehicle/chargeVehicle?licensePlate=971XDP")
        .contentType("application/json");
    this.server.perform(requestChargeVehicle);
    RequestBuilder requestVehicle = MockMvcRequestBuilders
        .get("/vehicle/consultVehicle?licensePlate=971XDP")
        .contentType("application/json");
    JSONObject jsonVehicle = new JSONObject(this.server.perform(requestVehicle).andReturn().getResponse().getContentAsString());
    assertTrue(Integer.parseInt(jsonVehicle.get("battery").toString()) == 100);
    assertTrue(Boolean.parseBoolean(jsonVehicle.get("available").toString()) == true);
}

private void addLowBatteryVehicle() throws Exception {
    JSONObject jsоЩehicle = new JSONObject()
        .put("type", "car")
        .put("licensePlate", "971XDP")
        .put("model", "Nissan GTR")
        .put("address", "C/Terreras, 9")
        .put("nSeats", 4);
    RequestBuilder requestAddVehicle = MockMvcRequestBuilders
        .post("/vehicle/addVehicle")
        .contentType("application/json")
        .content(jsоЩehicle.toString());
    this.server.perform(requestAddVehicle);
    RequestBuilder requestdischargeVehicle = MockMvcRequestBuilders
        .get("/vehicle/dischargeVehicle?licensePlate=971XDP")
        .contentType("application/json");
    this.server.perform(requestdischargeVehicle);
}
```

Pruebas unitarias Cargar Vehículo

```

@Test @Order(3)
void testRegisterTelephone() throws Exception{
    ResultActions OkeyData = this.sendRegister("antonioRe@gmail.com", "Antonio", "Perez Reverte", "12345678Z",
        CONTRASENA, ROLE_USER, CARNET, "612121212", BIRTHDATE);
    OkeyData.andExpect(status().isCreated());

    ResultActions WrongData1 = this.sendRegister("fernando@gmail.com", NAME, SURNAME, DNI,
        CONTRASENA, ROLE_USER, CARNET, "812121212", BIRTHDATE);
    WrongData1.andExpect(status().isConflict());

    ResultActions WrongData2 = this.sendRegister("daniel@gmail.com", NAME, SURNAME, DNI,
        CONTRASENA, ROLE_USER, CARNET, "612121212", BIRTHDATE);
    WrongData2.andExpect(status().isConflict());
}

@Test @Order(4)
void testRegisterEmail() throws Exception{
    ResultActions OkeyData = this.sendRegister("Antonio@gmail.com", "Antonio", "Perez Perez", "63548799Y",
        CONTRASENA, ROLE_USER, CARNET, PHONE_NUMBER, BIRTHDATE);
    OkeyData.andExpect(status().isCreated());

    ResultActions WrongData1 = this.sendRegister("fernandogmail.com", NAME, SURNAME, DNI,
        CONTRASENA, ROLE_USER, CARNET, PHONE_NUMBER, BIRTHDATE);
    WrongData1.andExpect(status().isConflict());

    ResultActions WrongData2 = this.sendRegister("daniel@gmailcom", NAME, SURNAME, DNI,
        CONTRASENA, ROLE_USER, CARNET, PHONE_NUMBER, BIRTHDATE);
    WrongData2.andExpect(status().isConflict());
}

```

Pruebas unitarias Registro Usuarios

```

@Test @Order(7)
void deactivateUser() throws Exception {
    ResultActions OkeyDeactivate = this.sendDeactivateUser("LuisDi@gmail.com");
    OkeyDeactivate.andExpect(status().isOk());

    ResultActions OkeyDeactivate1 = this.sendDeactivateUser("Izaskun@gmail.com");
    OkeyDeactivate1.andExpect(status().isOk());

    ResultActions WrongDeactivate = this.sendDeactivateUser("usuarioInexistente@gmail.com");
    WrongDeactivate.andExpect(status().isConflict());
}

@Test @Order(8)
void deleteUsers() throws Exception {
    String [] emails = {"Maria@gmail.com", "aurora@gmail.com", "Izaskun@gmail.com",
        "antonioRe@gmail.com", "Antonio@gmail.com", "pedro12@gmail.com", "LuisDi@gmail.com"};
    for(int i = 0; i < emails.length; i++) {
        JSONObject jsouser = new JSONObject()
            .put("email", emails[i]);
        RequestBuilder requestDelete = MockMvcRequestBuilders
            .put("/users/delete")
            .contentType("application/json")
            .content(jsouser.toString());
        this.server.perform(requestDelete).andExpect(status().isOk());
    }
}

```

Pruebas unitarias Desactivación Usuarios

3.3 TDD

Para conseguir un desarrollo basado en pruebas (TDD), que enfatice la escritura de pruebas automatizadas antes de implementar una funcionalidad real, los desarrolladores acordamos con el Product Owner un conjunto de historias de usuario las cuales teníamos que desarrollar a través de esta metodología:

- ❖ Durante el Sprint 1 (véase en 3.2.4):
 - Registro usuarios
 - Desactivación de usuarios
- ❖ Durante el Sprint 2 (véase en 3.2.4):
 - Reservar vehículos
 - Cargar vehículos

4 Desarrollo

4.1 Arquitectura del Proyecto

4.1.1 Arquitectura Frontend

En nuestro desarrollo se ha utilizado la arquitectura MVC (Model-View-Controller) que es un patrón de diseño. Es el utilizado por Angular. La arquitectura MVC en el contexto de nuestra aplicación de aplica así:

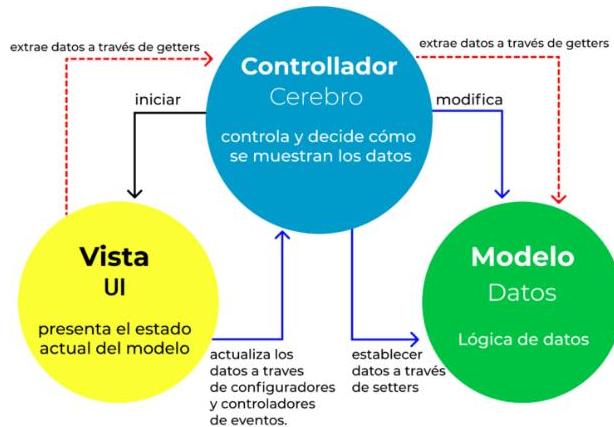
- **Model (Modelo):** Representado por clases TypeScript que definen la estructura y lógica de los datos.
 - **View (Vista):** Representada por las plantillas HTML y CSS mejoradas con características de Angular, que definen cómo se presenta la información al usuario.
 - **Controller (Controlador):** Representado por los componentes de Angular, que encapsulan la lógica de presentación y controlan la interacción entre la vista y el modelo.
- En nuestro caso, se han decidido crear los siguientes:
- Componente admin: en él residen todas las funcionalidades que están permitidas para el rol de administrador
 - Componente user: en él residen todas las funcionalidades que están permitidas para el rol de usuario.
 - Componente personal: en él residen todas las funcionalidades que están permitidas para el rol personal de mantenimiento.
 - Componente register: en él reside el registro de la aplicación para los usuarios.
 - Componente login: en él reside el login para los 3 roles de la aplicación.
 - Componente forgot-pwd: aglutina la acción ¿Olvidaste tu contraseña?, así como la introducción del correo electrónico para recuperarla.
 - Componente reset-pwd: engloba la introducción de una nueva contraseña tras recibir el correo de recuperación.
 - Componente page-not-found: en él reside una interfaz que se mostrará en caso de que el usuario introduzca una url no soportada por nuestra aplicación.
- **Services:** Los servicios son objetos que se utilizan por los diferentes componentes de la aplicación y se encargan de proveer datos y funcionalidades. Los servicios se han injectado en los componentes para utilizar sus métodos y funcionalidades. Éstos son los que permiten la comunicación frontend-backend.

En nuestra aplicación se ha decidido crear los siguientes:

- Admin.service: En él residen los métodos que se comunicarán con los controllers del backend relacionados con las funcionalidades de los administradores.
- Personal-service: En él residen los métodos que se comunicarán con los controllers del backend relacionados con las funcionalidades del personal de mantenimiento.
- User.service: En él residen los métodos que se comunicarán con los controllers del backend relacionados con las funcionalidades de los usuarios.
- Account.service: En él residen los métodos que se comunicarán con los controladores del backend relacionados con las funcionalidades de registro, login, recuperar contraseña.

Esta separación de responsabilidades nos ha facilitado la reutilización del código en el desarrollo frontend.

Patrones de Arquitectura MVC



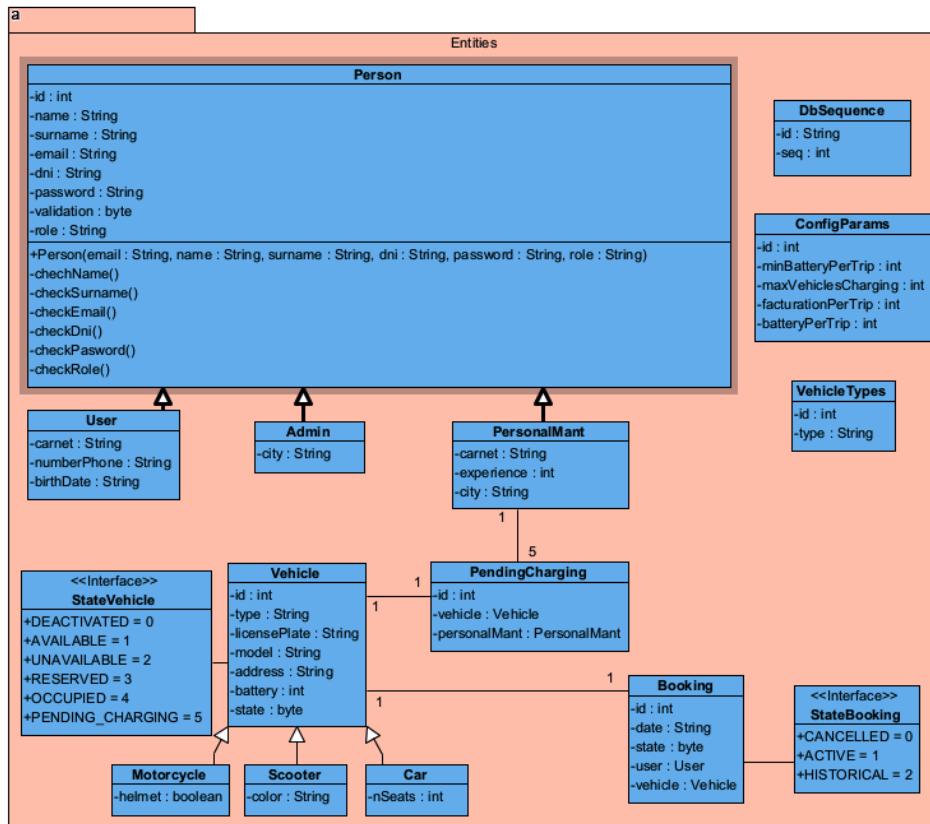
4.1.2 Arquitectura del Backend

En Java Spring usamos la arquitectura de tres capas o arquitectura MVC (Model-View-Controller) con un enfoque adicional en la capa de persistencia de datos.

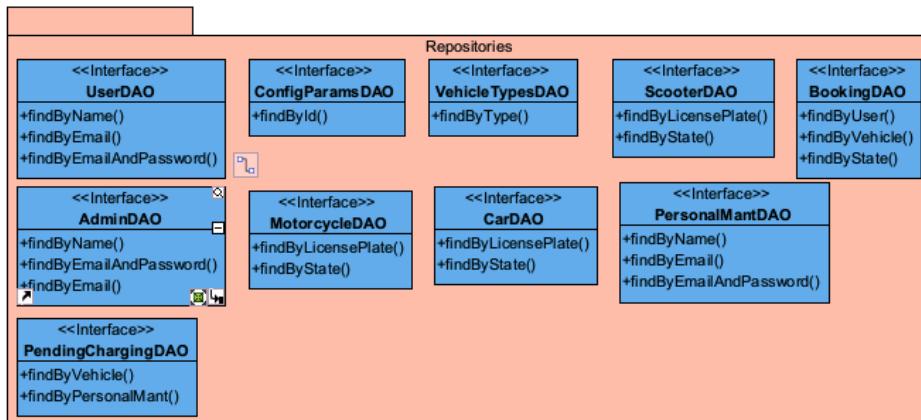
- **Controladores (Controllers):** Los controladores en Spring gestionan las solicitudes HTTP, interactúan con los datos del cliente y coordinan la lógica del negocio. Se encargan de recibir las solicitudes del cliente, invocar la lógica del negocio correspondiente y devolver las respuestas adecuadas.
- **Servicios (Services):** Los servicios contienen la lógica del negocio de la aplicación. Están diseñados para encapsular y manejar la funcionalidad específica de la aplicación que no debería estar directamente en los controladores. Los servicios se utilizan para dividir la lógica del negocio de los controladores y hacer que sea más modular y reutilizable.
- **Repositorios (Repositories):** Los repositorios en Spring se utilizan para interactuar con la capa de persistencia de datos. Los repositorios proporcionan métodos para realizar operaciones de lectura y escritura en la base de datos, abriendo los detalles de la implementación de la base de datos específica.

Esta arquitectura sigue el principio de separación de preocupaciones y promueve la modularidad y el mantenimiento del código. Ayuda a dividir la aplicación en componentes que son responsables de manejar diferentes aspectos, como la lógica del negocio y la persistencia de datos.

Para manejar los datos en la base de datos MongoDB, utilizamos la dependencia "**spring-boot-starter-data-mongodb**" para poder crear entidades dentro de nuestro código que se correspondan con las tablas (documentos en MongoDB) de la base de datos. La arquitectura seguida se ve reflejada en la siguiente imagen:



Para cada documento, debemos tener una interfaz que nos permita extraer datos de la base de datos.



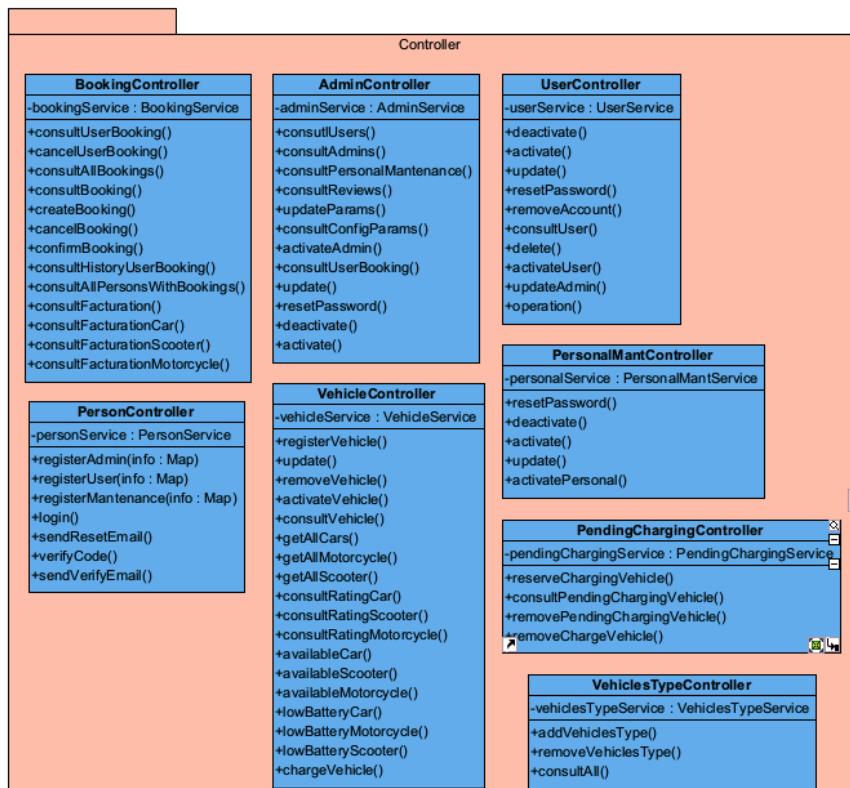
Pasando a la capa de negocio, disponemos de una serie de Servicios que implementan toda la lógica del sistema. Hemos optado por encapsular la funcionalidad de cada entidad dentro de un servicio.

Por ejemplo, si un administrador desea realizar cualquiera de su funcionalidad, los métodos encargados de gestionar esa lógica se encontrarán dentro del "AdminService". El problema lo encontramos cuando tenemos elementos que son partícipes del sistema, pero no usuarios, como son los vehículos o reservas.

La lógica se encuentra localizada de la misma manera que antes, pero aquí es donde es realmente necesaria la seguridad para restringir el acceso de los diferentes usuarios del sistema únicamente a su funcionalidad.



La última capa, que son los controladores, se encargan de recibir las peticiones HTTP, comprobar que son correctas y dejar que la capa de negocio siga su curso.



4.2 Seguridad y Gestión de Roles

4.2.1 Contraseña encriptada

La primera medida de seguridad consiste en guardar la contraseña de acceso de cada usuario del sistema encriptada para que, si alguien consigue acceder a nuestra base de datos, no pueda conocer los datos de acceso de nuestros usuarios. El algoritmo de encriptación usado es Bcrypt.

```
password: "$2a$10$oQDsLShGWjUihwH6fEwxJ.xuhuH.BKPeXFQsM8cB3pWE3AEllKgDe/"  
validation: 1  
role: "ROLE_USER,"
```

4.2.2 Doble Factor de Autenticación

El 2FA implementado en la aplicación fue mediante Google Authenticator. Este está disponible únicamente para los usuarios de la misma. Tras un registro correcto, deberán escanear un código QR desde su dispositivo móvil e introducir un código de 6 cifras que la aplicación le proporcionará.



Posteriormente, cada vez que un usuario quiera iniciar sesión, deberá introducir el código proporcionado por la aplicación.



4.2.3 JSON Web Token (JWT)

Hemos implementado un token, que cuando un usuario inicia sesión (en caso de administrador y mantenimiento, lo recibe tras un correcto inicio de sesión y un usuario normal tras pasar el 2FA) recibe un token acorde a su rol permitiendo acceder a ciertos recursos, siguiendo el patrón RBAC.

```

▼ token:Array
  0:"eyJhbGciOiJIUzI1NiJ9"
  1:"eyJzdWlOiJhbGVqYW5kcm8ubWVkaW5hN...TM3MTU2MCwiZXhwIjoxNzAxMzc0MjYwfQ"
  2:"5kqysVRfXJJPQU5TnKo8IHYZsKGJ2VVyyWPSASPF4"
  length:3
  ...
  
```

Todas las reglas de acceso están configuradas en la clase SecurityConfig.java del backend. Aquí se muestran algunas:

```

authorizeHttpRequests(authRequest ->
    authRequest
        .requestMatchers("/admins/*").hasRole("ADMIN")
        .requestMatchers("/bookings/createBooking").hasRole("USER")
        .requestMatchers("/bookings/cancelUserBooking").hasRole("USER")
        .requestMatchers("/bookings/confirmBooking").hasRole("USER")
        .requestMatchers("/bookings/consultUserBooking").hasRole("USER")
        .requestMatchers("/bookings/consultHistoryUserBooking").hasRole("USER")
        .requestMatchers("/bookings/consultAllPersonsWithBookings").hasRole("ADMIN")
        .requestMatchers("/bookings/consultBooking").hasRole("ADMIN")
        .requestMatchers("/bookings/consultFacturation").hasRole("ADMIN")
        .requestMatchers("/bookings/consultFacturationCar").hasRole("ADMIN")
        .requestMatchers("/bookings/consultFacturationScooter").hasRole("ADMIN")
    )
  
```

Si un usuario intenta acceder a un recurso no autorizado, recibirá como respuesta un 403 FORBIDDEN.

4.3.4 Correo de verificación

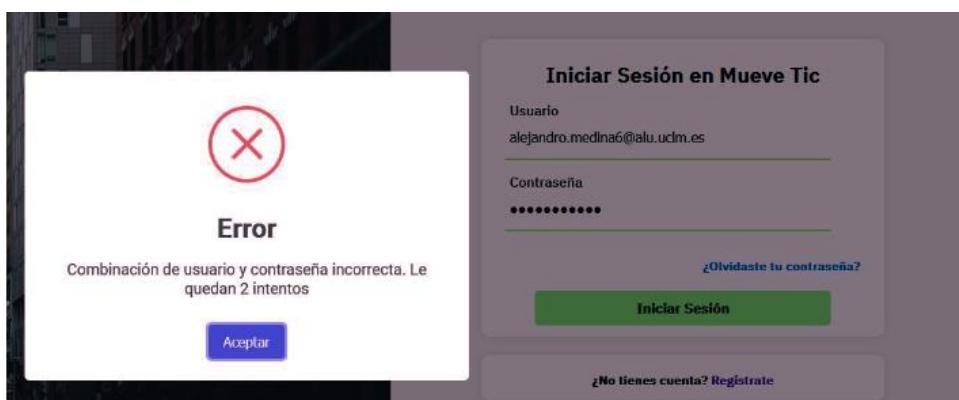
Tras el registro de un usuario, este recibe un correo electrónico para verificar su dirección. Cuando el usuario activa su cuenta haciendo clic en el correo, mandamos esta petición al backend para la activación del usuario:

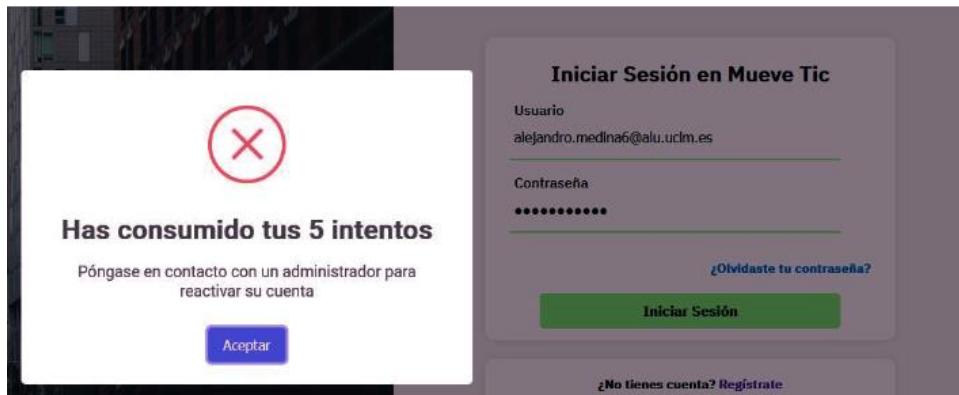
<https://muevetic.onrender.com/user/activate?email=5R4e8iCEf2sVHbIE7MVVoXrguInnbDfHKJGHzM4ihK8g=>

Utiliza un método de encriptación simétrico **AES** para encriptar y desencriptar.

4.3.5 Bloqueo cuenta tras 5 intentos de inicio de sesión.

Cuando un usuario falla 5 veces el inicio de sesión, su cuenta se bloquea.





Para volver a recuperar la cuenta, el usuario deberá ponerse en contacto con un administrador.

Logs del backend:

```
Resolved [org.springframework.web.server.ResponseStatusException: 409 CONFLICT "4"]
Resolved [org.springframework.web.server.ResponseStatusException: 409 CONFLICT "3"]
Resolved [org.springframework.web.server.ResponseStatusException: 409 CONFLICT "2"]
Resolved [org.springframework.web.server.ResponseStatusException: 409 CONFLICT "1"]
Resolved [org.springframework.web.server.ResponseStatusException: 423 LOCKED "User has been blocked"]
```

4.3.6 Guardas de Seguridad en Angular

En Angular disponemos de guardas que permiten restringir el acceso a determinadas URL. Podemos indicar el criterio de aceptación para el acceso creando diversas guardas de angular.

```
3)
export class AdminSecurity implements CanActivate {
    constructor(private accountService: AccountService, private router: Router) { }

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): boolean {
        if (sessionStorage.getItem('role') === 'ROLE_ADMIN' && sessionStorage.getItem('token')) {
            return true;
        }
        sessionStorage.clear();
        this.router.navigate(['/login']);
        return false;
    }
}
```

En caso de que no se cumplan los requisitos, se redirigirá a la pantalla de login.

Para aplicarlas, es necesario indicarlo en el archivo: **"app-routing.module.ts"**.

```
const routes: Routes = [
    { path: '', redirectTo: 'login', pathMatch: 'full' },
    { path: 'login', component: LoginComponent },
    { path: 'register', component: RegisterComponent },
    { path: 'forgot-password', component: ForgotPwdComponent },
    { path: 'reset-pwd', component: ResetPwdComponent },
    { path: 'user', component: UserComponent, canActivate: [AuthGuard, UserSecurity] },
    { path: 'admin', component: AdminComponent, canActivate: [AuthGuard, AdminSecurity] },
    { path: 'personal', component: PersonalComponent, canActivate: [AuthGuard, PersonalSecurity] },
    { path: '**', component: PageNotFoundComponent },
];
```

4.3 Calidad

A continuación, se detalla el procedimiento seguido para conseguir la calidad requerida tanto en Frontend como en Backend.

4.3.1 Calidad en Frontend

Se ha conseguido cumplir con todos los requisitos de calidad pedidos por el stakeholder de calidad. Excepto el requisito “Lograr una cobertura de código del 60% o superior y evitar la existencia de código no utilizado”, ya que se decidió con el stakeholder eliminar este requisito en Frontend tras no ser necesario realizar pruebas unitarias.

A continuación, se detallan los demás requisitos de calidad y las acciones tomadas para cumplir con ellos.

- Gestión de vulnerabilidades:
 - Objetivo: Lograr un software con el mínimo de vulnerabilidades, identificándolas y corrigiéndolas de manera adecuada.
 - Acciones Tomadas:
 - Utilización de herramientas como Sonar y Sonar Lint para analizar e identificar vulnerabilidades en el código.
 - Para ello, antes de comenzar el desarrollo del proyecto se decidió por unanimidad las diferentes reglas de sonarlint que aplicarían a nuestro proyecto, eliminando las no necesarias, así como añadiendo las faltantes. Se revisó y aprobó por el stakeholder de calidad.
 - Se definieron los quality gates en acuerdo común con el equipo y el stakeholder de calidad que aplicarían al proyecto Frontend.
- Reducción de Duplicación de Código
 - Objetivo: Mantener la duplicación de código inferior al 10%.
 - Acciones Tomadas:
 - Revisión periódica del código para identificar y eliminar duplicaciones.
 - Utilización de herramientas de análisis estático para detectar duplicación y aplicar refactorización cuando sea necesario (GitHub Copilot).
- Integración Continua
 - Objetivo: Automatizar el análisis de código con SonarQube para garantizar calidad constante y entrega continua.
 - Acciones Tomadas:
 - Configuración de stage “Sonarqube” en el pipeline creado para proyecto Frontend. Contiene steps y jobs en los que se ejecuta sonarqube mediante integración continua.
- Estabilidad en Sprints Posteriores
 - Objetivo: Mantener o mejorar los resultados de calidad alcanzados en el primer Sprint en los siguientes.
 - Acciones Tomadas:
 - Monitoreo continuo de métricas de calidad en sprints subsiguientes.
 - Implementación de medidas correctivas para mantener o mejorar la calidad del código.
- Herramientas de Verificación

- Objetivo: Utilización de herramientas Sonar y Sonar Lint para verificar el cumplimiento de los criterios de calidad.
- Acciones Tomadas:
 - Integración de herramientas de análisis estático en el flujo de trabajo de desarrollo.
 - Revisión regular de informes generados por SonarQube y aplicación de correcciones según ha sido necesario.

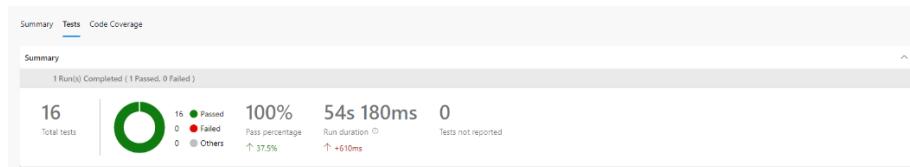
4.3.2 Calidad en Backend

Al igual que en el frontend, también se ha conseguido cumplir con todos los requisitos de calidad pedidos por el stakeholder de calidad. A diferencia del anterior, el requisito “Lograr una cobertura de código del 60% o superior y evitar la existencia de código no utilizado”, fue acordado entre los involucrados en su reducción al 10%.

Además, tras el primer sprint se acordó con el stakeholder de calidad y product owner la eliminación en el quality gate creado de “complejidad ciclomática” al encontrar incompatibilidades de versión sonar y no identificarlos adecuadamente. Aunque el equipo se comprometió durante todo el proyecto a evitar la creación de métodos con una complejidad ciclomática excesiva.

El resto de los requisitos no funcionales que nos hacen cumplir con la calidad son:

- Seguridad de Datos:
 - Objetivo: Proteger la integridad y confidencialidad de los datos.
 - Acciones Tomadas:
 - Implementación de protocolos de seguridad, como encriptación.
- Integración Continua
 - Objetivo: Automatizar el análisis de código con SonarQube para garantizar calidad constante y entrega continua.
 - Acciones Tomadas:
 - Generación de gráficos de visualización de test: para evaluar el rendimiento del código.



- Adaptación de pipeline para poder generar de manera automática el control de los quality gates.
- Realización de pruebas unitarias:
 - Objetivo: El objetivo fundamental de las pruebas unitarias en el backend es asegurar la correcta funcionalidad de las unidades de código a nivel de métodos y funciones. Estas pruebas buscan validar el comportamiento individual de cada componente para garantizar su correcto funcionamiento, identificar posibles errores y contribuir a la creación de un sistema robusto y confiable.

- Acciones Tomadas:
 - Selección del Framework de Pruebas: Se optó por utilizar JUnit como el framework principal para la creación y ejecución de pruebas unitarias en el backend. JUnit es ampliamente reconocido por su facilidad de uso y robustez en la ejecución de pruebas en entornos Java.
 - Definición de Casos de Prueba: Se identificaron las unidades de código críticas en el backend que requerían pruebas unitarias exhaustivas. Para cada una de estas unidades, se definieron casos de prueba que cubrían diferentes escenarios, incluyendo casos límite y situaciones de error.
 - Implementación de Pruebas con JUnit: Se crearon suites de pruebas utilizando JUnit para las diversas unidades de código en el backend. Cada prueba se diseñó para verificar un aspecto específico del comportamiento del código, asegurando una cobertura significativa.
 - Integración con Jacoco para Análisis de Cobertura: Se integró Jacoco como herramienta para medir la cobertura de código lograda por las pruebas unitarias. Jacoco proporciona informes detallados sobre qué porcentaje del código está cubierto por las pruebas, permitiendo identificar áreas que requieren una mayor atención.
 - Establecimiento de Objetivos de Cobertura: Se establecieron objetivos específicos de cobertura de código que debían alcanzarse con las pruebas unitarias. Aunque se acordó reducir el requisito inicial al 10%, se trabajó para garantizar una cobertura exhaustiva, abarcando tanto casos normales como excepcionales.
 - Análisis de Resultados de Pruebas: Se revisaron regularmente los resultados de las pruebas unitarias, utilizando informes generados por JUnit y Jacoco. Se identificaron y corrigieron posibles errores descubiertos durante las pruebas, asegurando la estabilidad del backend.
 - Integración con el Proceso de Integración Continua: Se configuró el proceso de integración continua para ejecutar automáticamente las pruebas unitarias en cada cambio realizado en el código fuente del backend. Esto aseguró una retroalimentación inmediata sobre la calidad del código.
 - Refactorización y Mejora Continua: Se realizó una refactorización continua del código en función de los resultados y hallazgos de las pruebas unitarias. Esto permitió mejorar la calidad del código, reducir posibles vulnerabilidades y mantener un alto estándar de robustez.

4.4 Limitaciones Encontradas

La principal limitación que nos hemos encontrado ha sido la base de datos. Al utilizar una base de datos no relacional, la cual no teníamos experiencia previa, tuvimos que hacer algunos ajustes e invertir tiempo para poder utilizarla de forma relacional.

Algunas funcionalidades que podrían resolverse fácilmente a nivel de base de datos con una consulta, en Mongo no hemos encontrado la manera de hacerlo, por lo tanto, hemos tenido que resolverlo en la capa de negocio aumentando la complejidad del sistema de manera innecesaria. Otro aspecto negativo de Mongo es que dispone de poca seguridad para nuestros datos, ya que no se puede realizar el proceso de enmascaramiento como en SQL Server ni hemos podido realizar auditorías.

5 Manual de Usuario

Proceso de Registro Usuario:

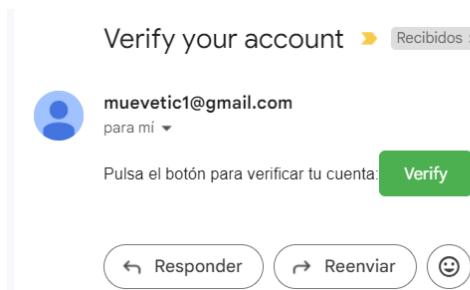
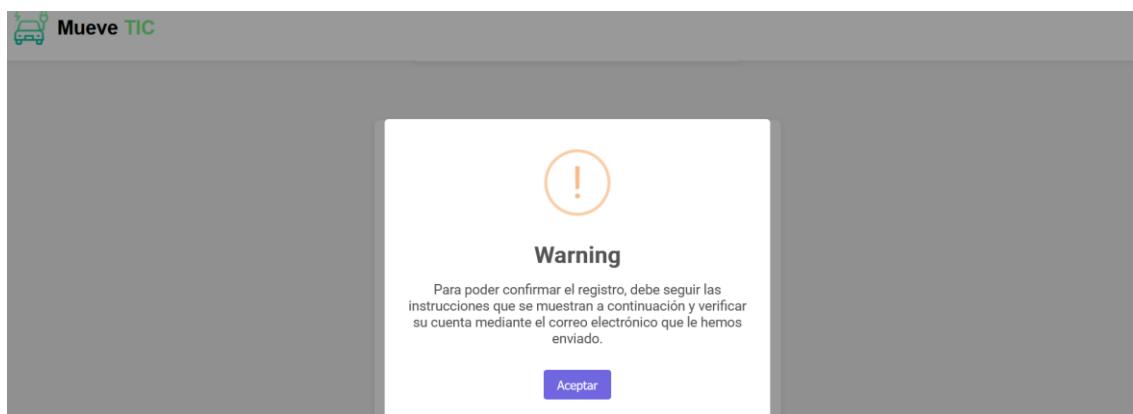
A la hora de realizar el registro en MueveTIC, deberemos de introducir los siguientes parámetros, donde aquellos que estén marcados con * son obligatorios para poder realizar el registro:

- **Nombre y dos Apellidos**
- **Email:** se valida si tiene un formato válido comprobando si contiene @.
- **Contraseña:** se valida si tiene una longitud mínima de ocho caracteres donde haya una mayúscula, una minúscula, un número y un carácter especial.
- **Repetir Contraseña:** se verifica que se introduce la misma para evitar errores por parte de usuario teniéndola que introducir otra vez a mano ya que se deshabilita la opción de pegar el texto.
- **DNI:** se comprueba si es válido si tiene una longitud mínima de 8 dígitos y una letra.
- **Móvil:** se comprueba que tenga una longitud mínima de 9 dígitos.
- **Fecha de Nacimiento:** se comprueba de que el usuario tiene más de 16 años.
- **Carnet de Conducir:** selección entre los tipos de carnets de A1, A2 y B

Crea tu cuenta

Nombre *	Alejandro
Apellidos *	Sánchez Arcos
Email *	alexsanchezamarillo@gmail.com
Contraseña *	*****
Repite Contraseña *	*****
DNI *	71358291D
Móvil *	607962433
Fecha de Nacimiento	4 3 2002
Carnet de conducir	B
Registrarse	

Una vez introducido los parámetros del registro tendremos que verificar el email, siguiendo el enlace del correo que la plataforma nos ha enviado:



Le damos al botón de verificar y deberemos de escanear el código QR en Google Authenticator e introducir el código:



Una vez verificado que el código es correcto podremos iniciar sesión.

Proceso de Iniciar Sesión:

Para iniciar sesión introduciremos nuestro correo y contraseña:

Iniciar Sesión en Mueve Tic

Usuario
alex.sanchezamarillo@gmail.com

Contraseña

[¿Olvidaste tu contraseña?](#)

Iniciar Sesión

[¿No tienes cuenta? Regístrate](#)

Una vez le demos a iniciar sesión, el doble factor nos pedirá un nuevo código:

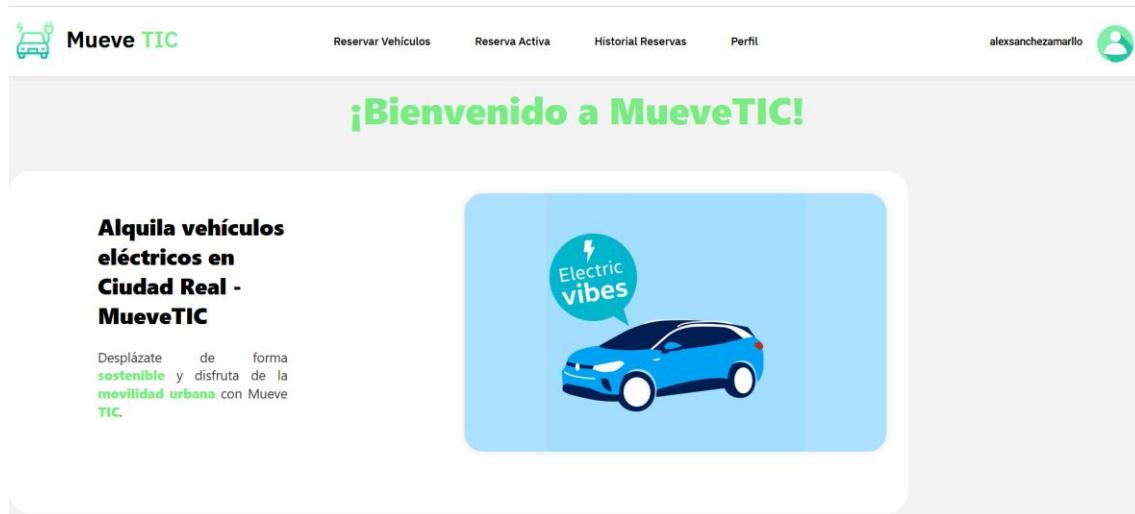
Segundo Factor de Autenticación

Introduce los 6 dígitos generados por la App Google Authenticator

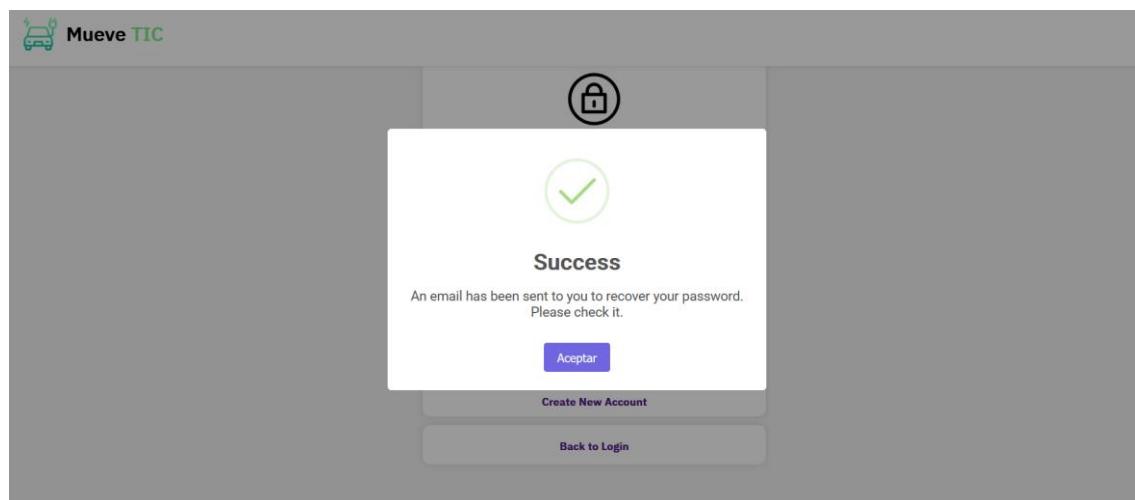
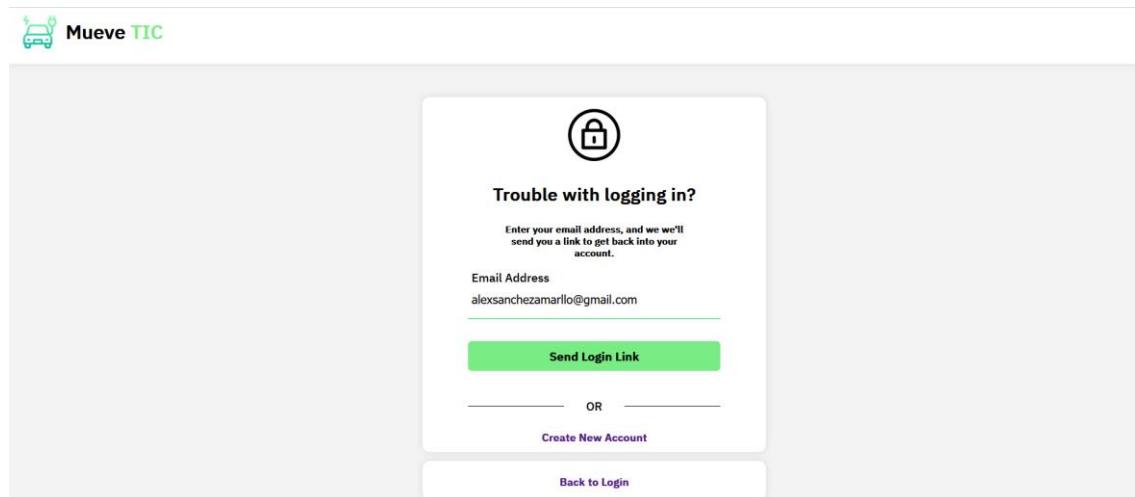
665779

Verificar Código

Según el rol que tenga asociado el usuario aparecerá una interfaz u otra, en este caso aparecería la interfaz de usuario:



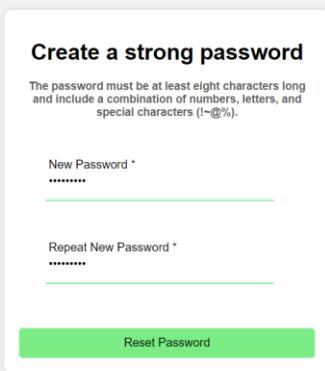
Proceso de Recuperar Contraseña:



Reset your account ➔ Recibidos x

 [muevetic1@gmail.com](#)
para mí ▾

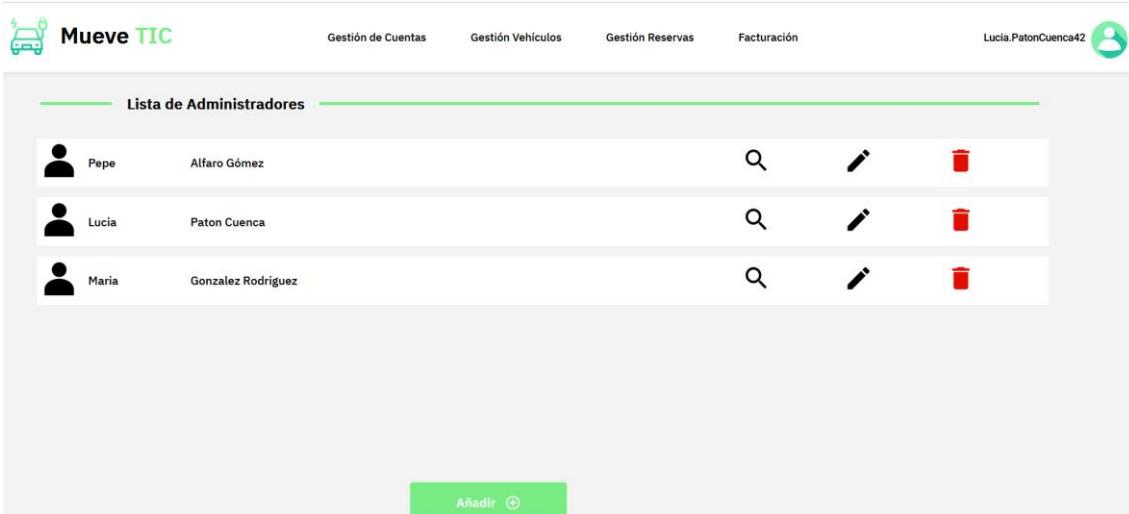
Pulsa el botón para resetear tu contraseña: [Reset Password](#)



ROL ADMINISTRADOR

Proceso Gestión Cuentas Administrador:

Iniciamos sesión con un administrador, en este caso, Lucia.PatonCuenca42 y seleccionamos en el menú superior central **Gestión de Cuentas**, y en el desplegable que nos aparece elegimos el tipo de usuario Administrador (IGUAL PARA USUARIO Y PERSONAL DE MANTENIMIENTO):



Lista de Administradores		
	Pepe	Alfaro Gómez
	Lucia	Paton Cuenca
	Maria	Gonzalez Rodriguez

Como podemos observar en la interfaz, nos aparece un listado de administradores donde podremos seleccionar que tipo de acción queremos hacer sobre el administrador elegido (Consultar, Modificar o Deshabilitar):

- ❖ Si pulsamos en la opción de **Consultar** (Lupa):

Mueve TIC

Gestión de Cuentas Gestión Vehículos Gestión Reservas Facturación

Lucia.PatonCuenca42

Consultar

Nombre: Maria
Apellidos: Gonzalez Rodriguez
Correo Electrónico: Maria.Rodriguez@hotmail.com

DNI: 04690936V
Ciudad: Albacete

Volver

Nos aparece la información de ese Administrador y los campos no podremos seleccionarlos ya que están deshabilitados.

- ❖ Si pulsamos sobre la opción de **Modificar** (Lápiz):

Mueve TIC

Gestión de Cuentas Gestión Vehículos Gestión Reservas Facturación

Lucia.PatonCuenca42

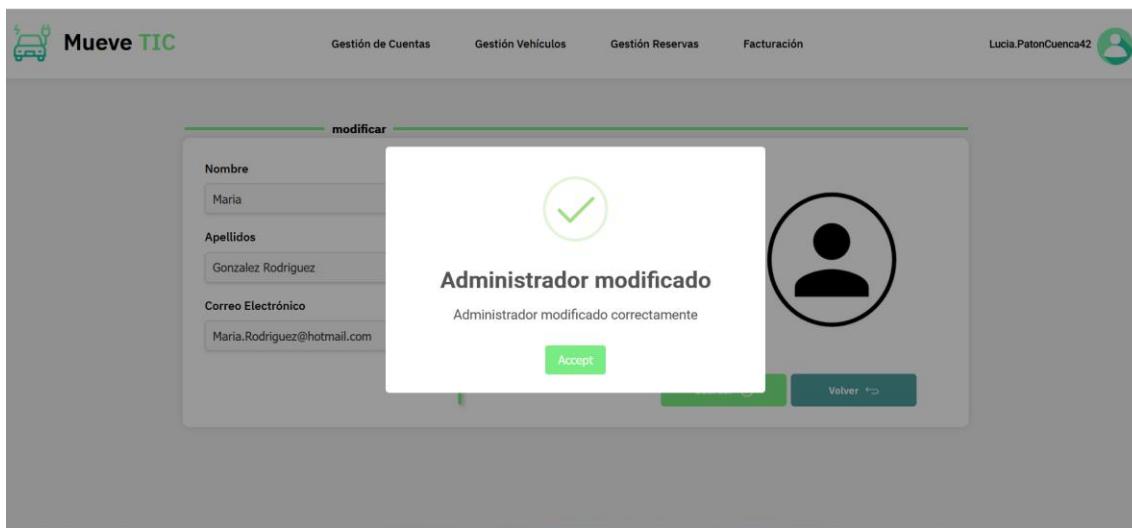
modificar

Nombre: Maria
Apellidos: Gonzalez Rodriguez
Correo Electrónico: Maria.Rodriguez@hotmail.com

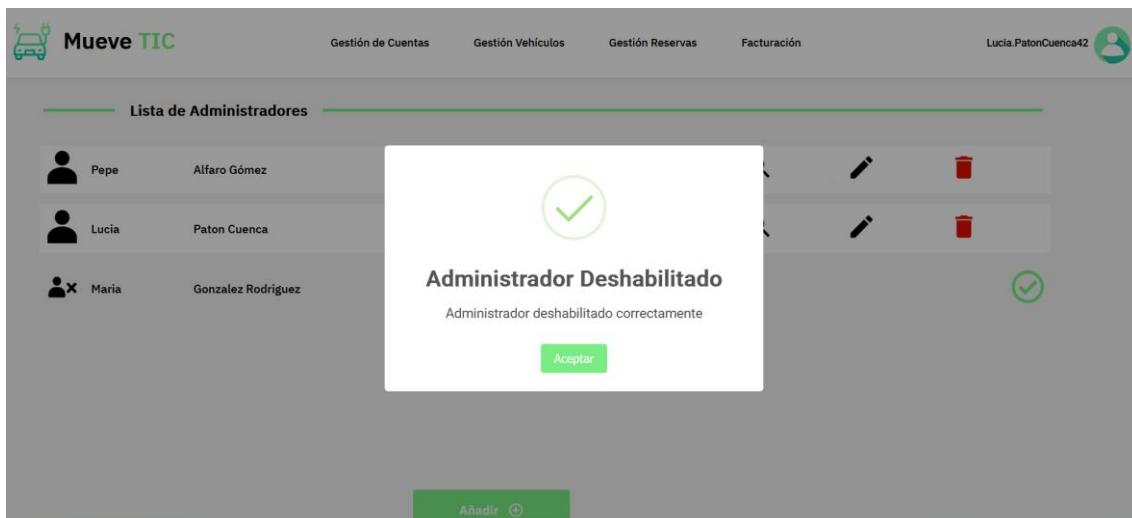
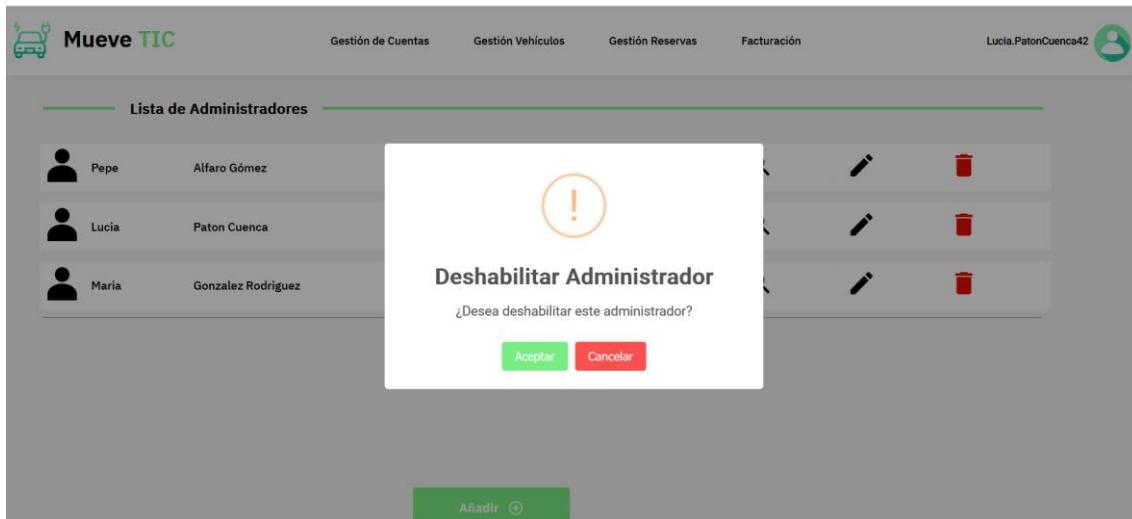
DNI: 04690936V
Ciudad: Albacete

Guardar Volver

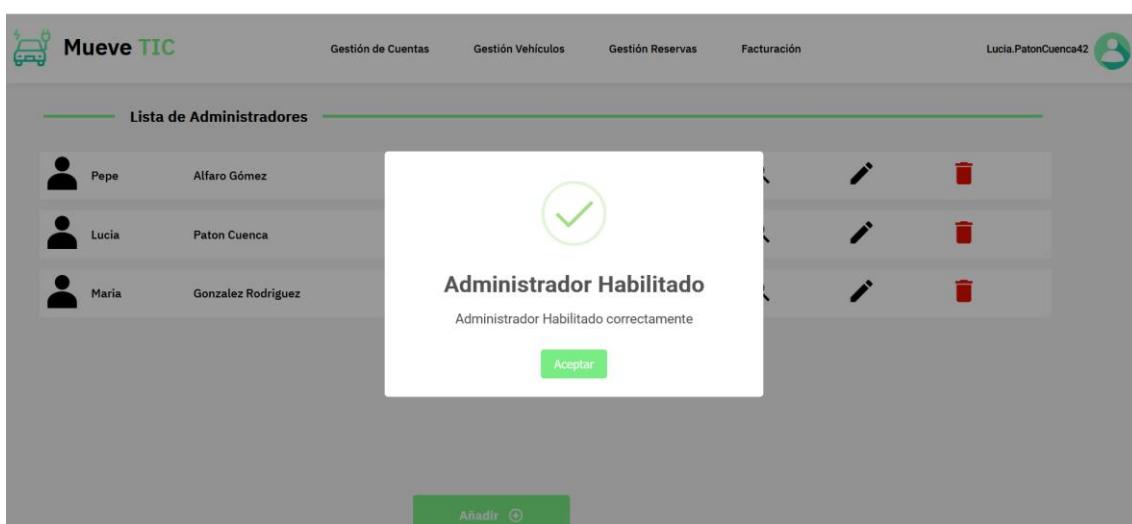
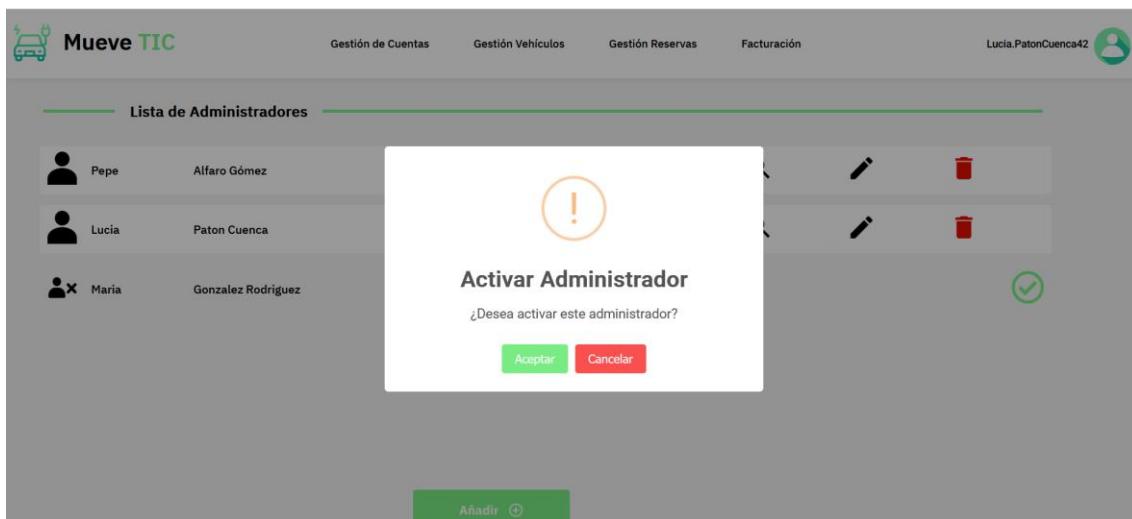
Como podemos observar ahora si podemos modificar alguno de los campos y para guardar los cambios pulsamos en el botón de guardar:



- ❖ Si pulsamos en la opción de **Deshabilitar** (Papelera):



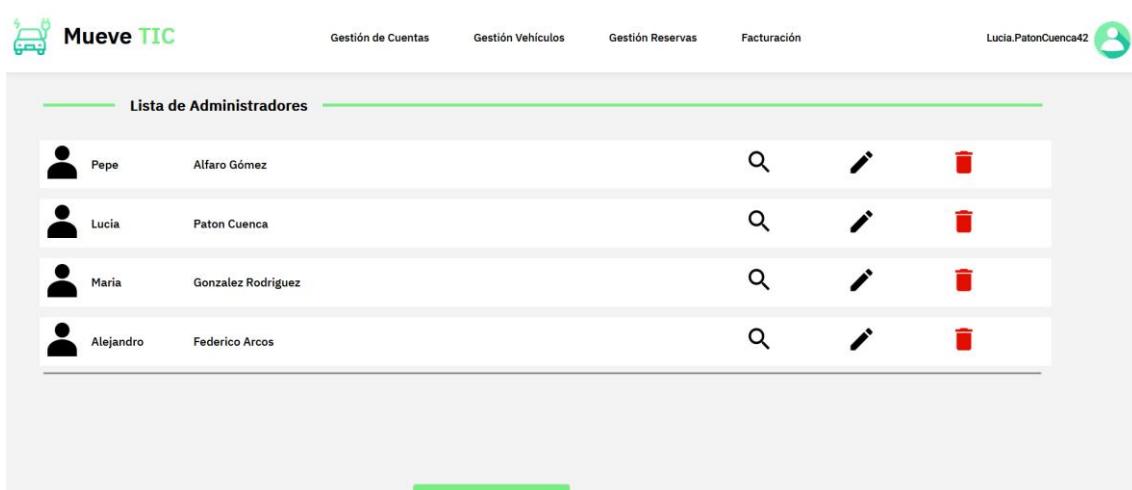
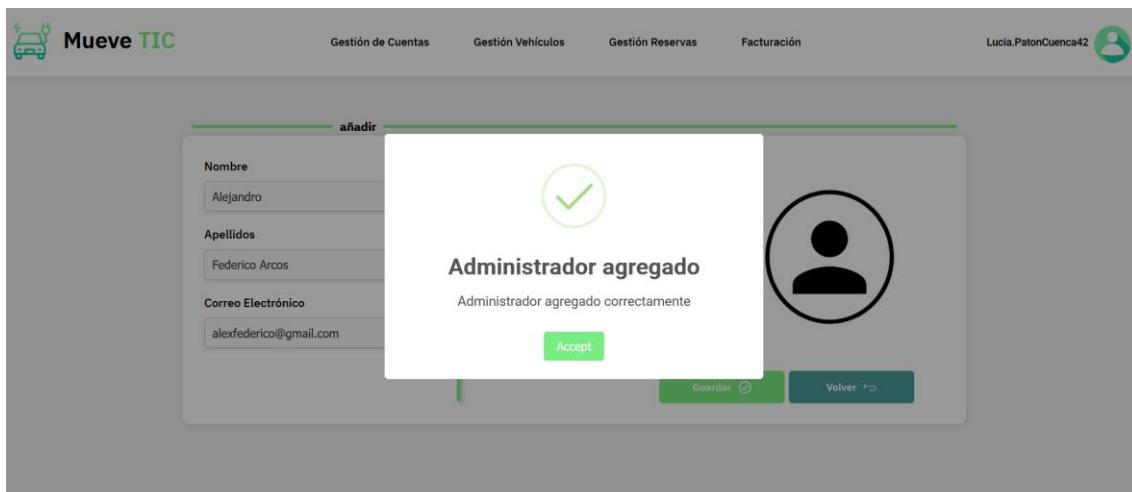
Como podemos observar en la interfaz ya no podemos realizar ninguna opción anterior sobre ese usuario y solo podremos volver a **Habilitarlo** dándole al ícono de Activar:



❖ Si pulsamos en la opción de **Añadir**:

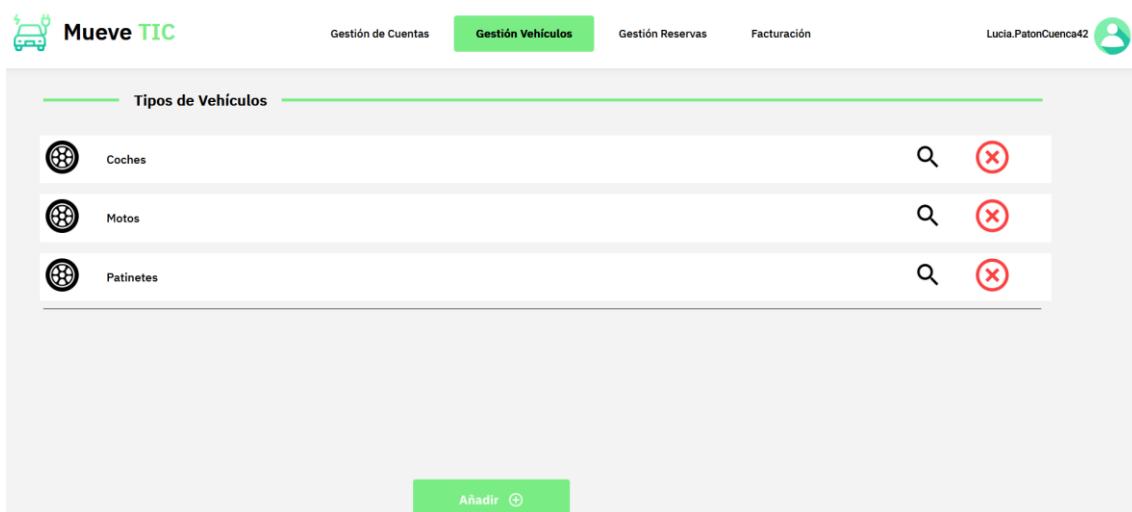
Introducimos la información del administrador que queramos crear y pulsamos en Guardar:

Nombre	DNI
Alejandro	71358293D
Apellidos	Ciudad
Federico Arcos	Ciudad Real
Correo Electrónico	Contraseña
alexfederico@gmail.com	*****



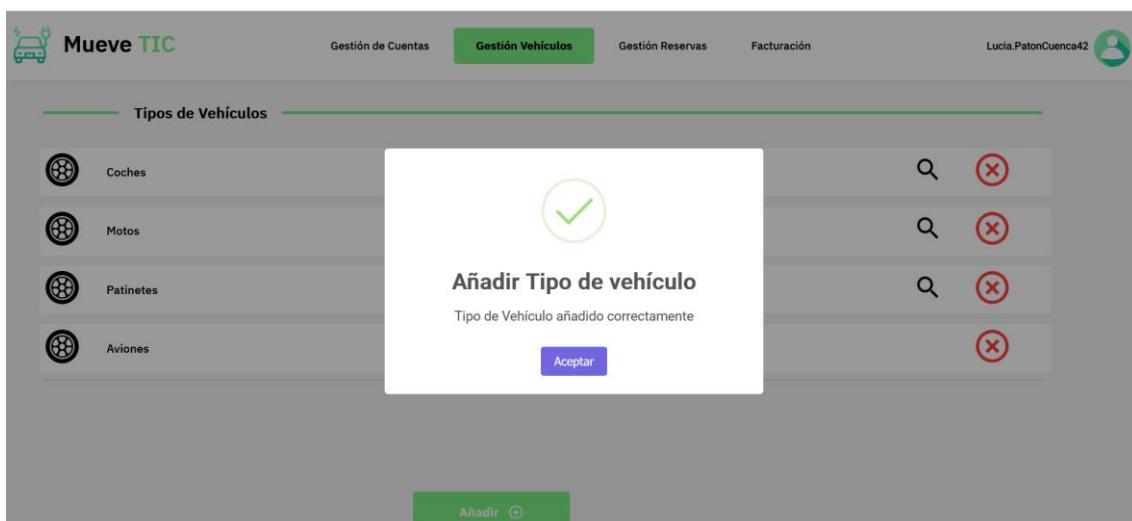
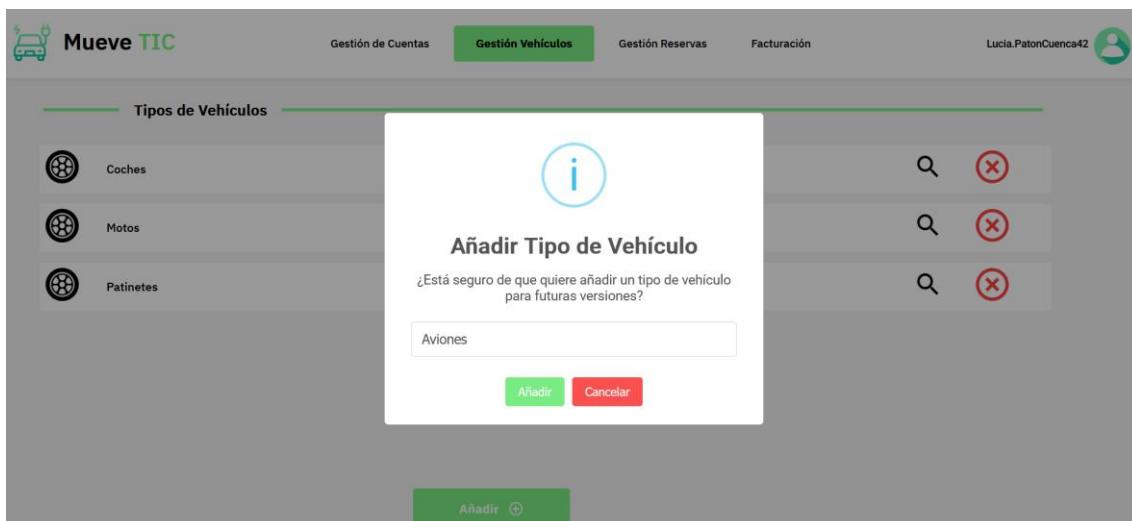
Proceso Gestión Vehículos Administrador:

Para poder gestionar los vehículos de la plataforma seleccionaremos la opción de **Gestionar Vehículos** en el menú superior central:

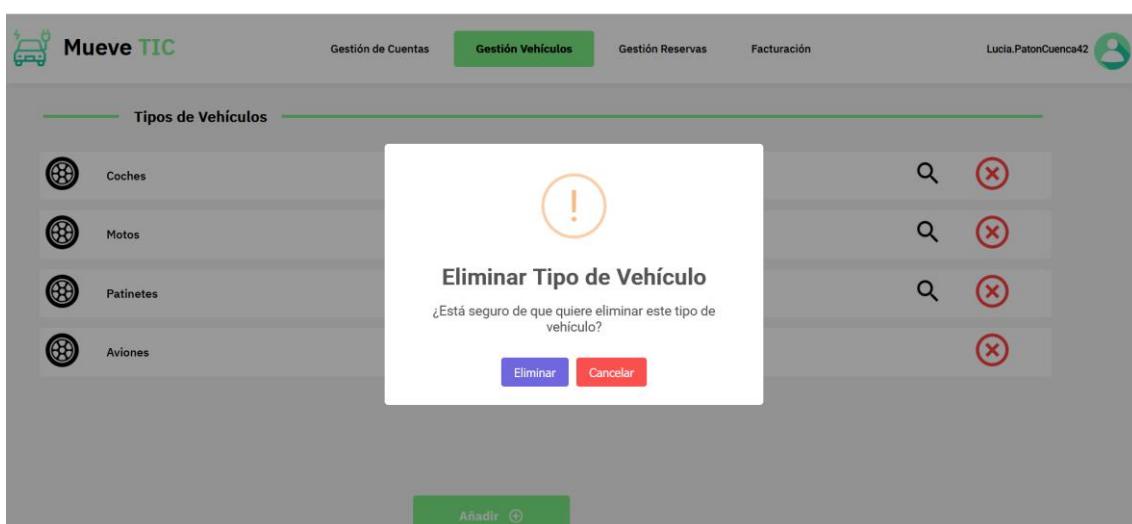


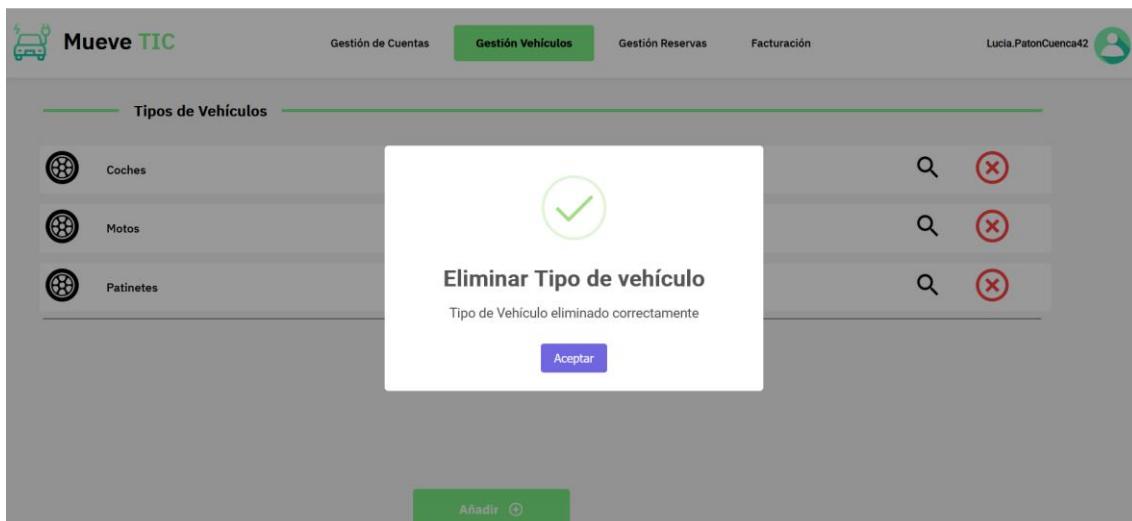
Como podemos observar en la interfaz, nos aparecen los tipos de vehículos disponibles en la plataforma donde podremos realizar tres acciones (Consultar, Eliminar y Añadir):

- ❖ Si pulsamos en **Añadir Tipo de Vehículo**:



- ❖ Si pulsamos en **Eliminar Tipo de Vehículo**:





- ❖ Si pulsamos en **Consultar Tipo de Vehículo**, en este caso, COCHE (SERÍA IGUAL CON MOTOS Y PATINETES):

Lista de Coches	
Peugeot 206	1234SAD
Mercedes Clase A	4569FGH
Audi TT	1237HJK
Seat León	9876RTY
Audi Q3	1238SDF

Como podemos observar en esta interfaz nos aparecen todos los coches de la plataforma con su valoración media. Podemos realizar las acciones, en este caso, de Consultar Coche, Modificar Coche, Deshabilitar Coche y Añadir Coche:

- ❖ Si pulsamos en **Consultar Coche**:

CONSULTAR

Modelo Peugeot 206	Matrícula 1234SAD
Número de Plazas 5	Estado Disponible
Ubicación C/Terreras, 9	Porcentaje de Batería 60

Volver ↺

Nos aparecerá la información del coche seleccionado.

- ❖ Si pulsamos en **Modificar Coche**:

MODIFICAR

Modelo Peugeot 206	Matrícula 1234SAD
Número de Plazas 5	Estado Disponible
Ubicación C/García, 10	Porcentaje de Batería 60

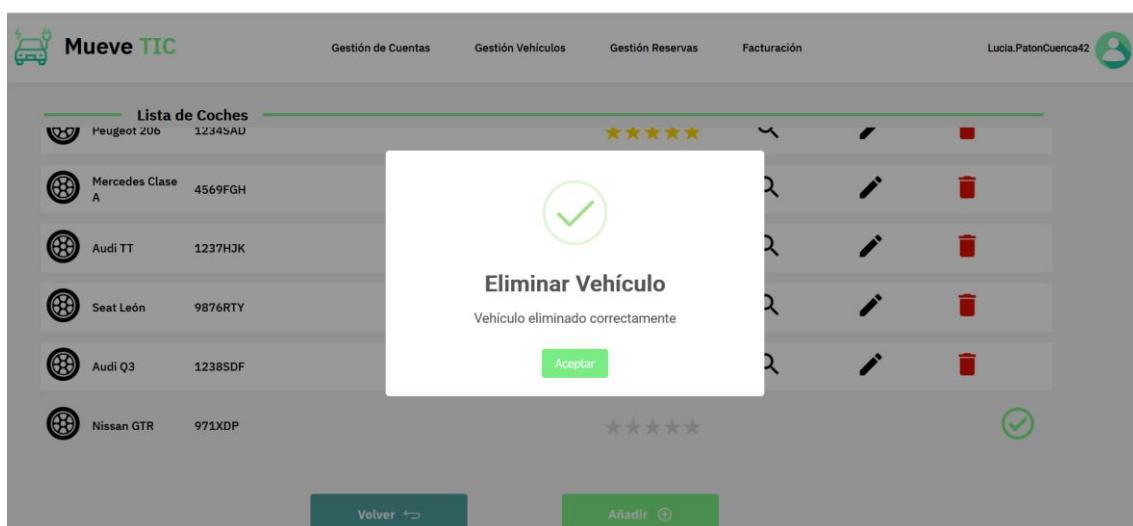
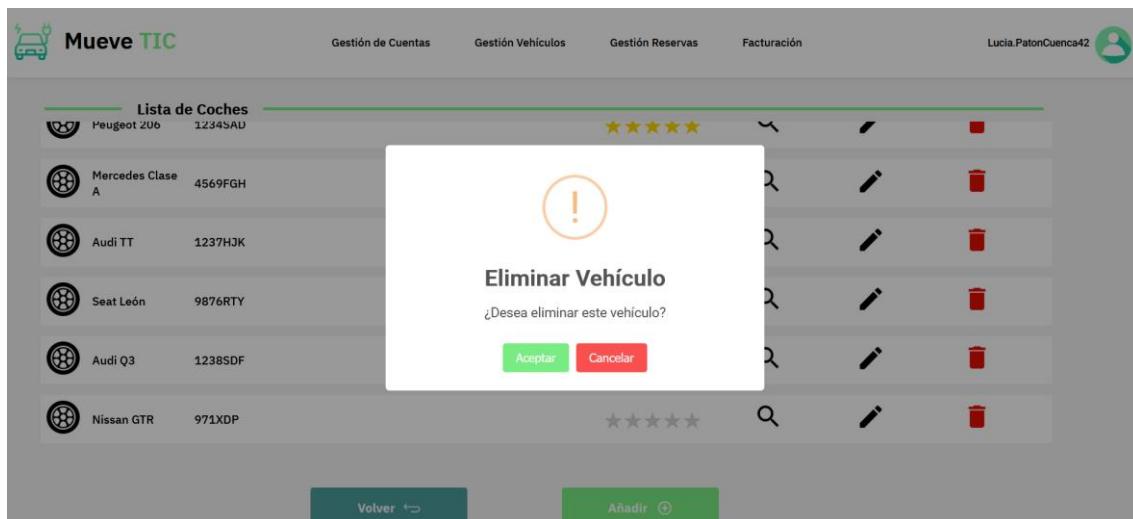
Guardar ⏮ Volver ↺

Si modificamos por ejemplo la ubicación del coche:

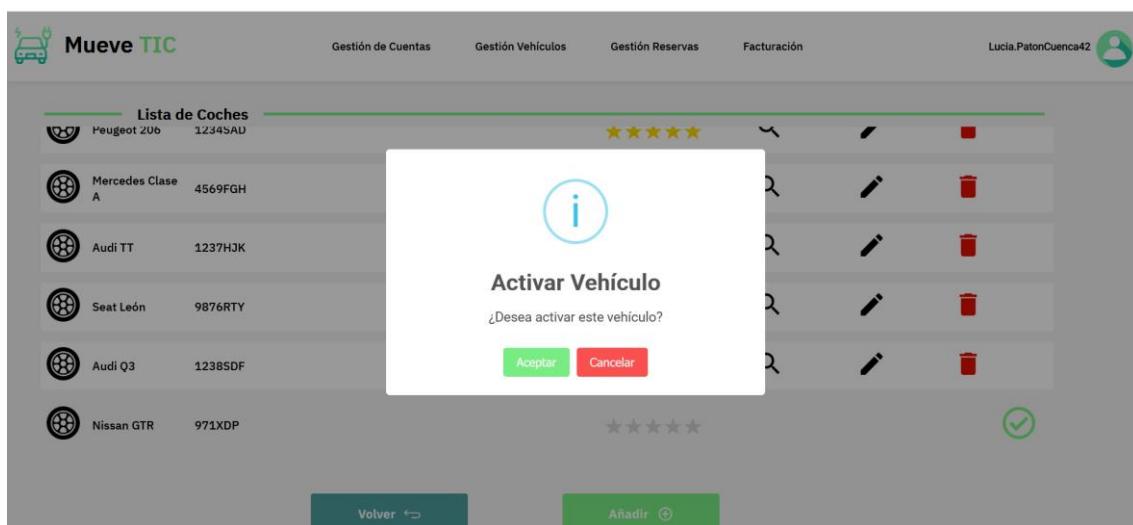
MODIFICAR

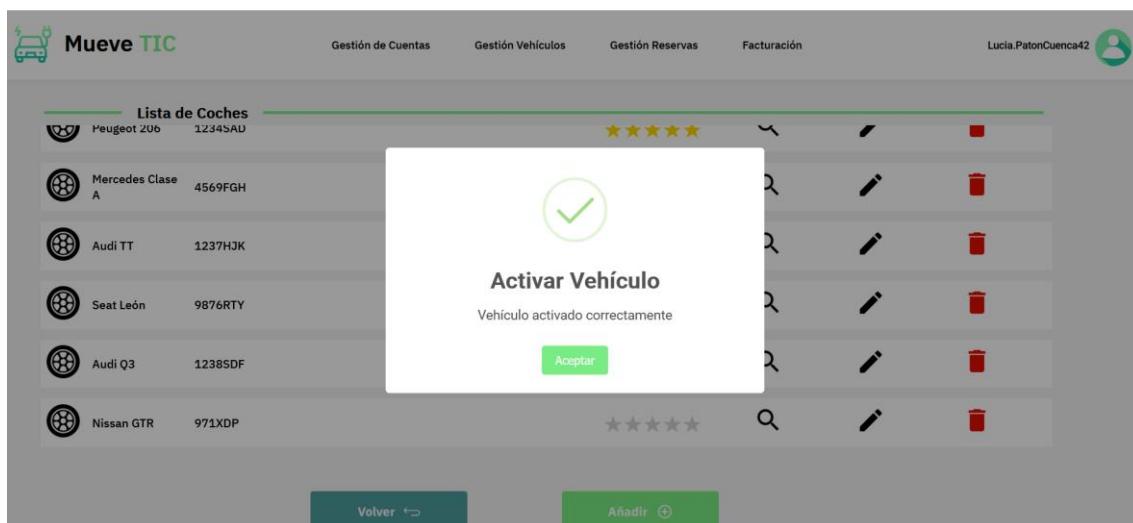
Modelo Peugeot 206	Actualizar Vehículo
Número de Plazas 5	Vehículo actualizado correctamente
Ubicación C/García, 10	Aceptar Volver ↺

- ❖ Si pulsamos en **Deshabilitar Coche**:



Como podemos observar, la única acción que podremos realizar sobre el Nissan GTR será **Activarlo** pulsando en el ícono de Activar:





❖ Si pulsamos en la opción **Añadir Coche**:

Introducimos los datos del coche y pulsamos en Guardar:

The screenshot shows an 'AÑADIR' (Add) form with fields for Modelo (Ferrari), Matrícula (1234CDG), Número de Plazas (3), Estado (Disponible), Ubicación (C/García,9), and Porcentaje de Batería (100). On the right is a large car wheel icon. Below the form is a modal with a green checkmark icon and the message 'Vehículo añadido correctamente' (Vehicle added correctly).

Proceso Gestión Reservas Administrador:

Para que el administrador pueda gestionar las reservas navegará y seleccionará la opción del menú superior central de **Gestión Reservas**:

Podrá consultar **Reservas por Usuario**:

Al seleccionar esa opción le aparecerá la siguiente interfaz:

The screenshot shows a user interface titled "Mueve TIC". At the top, there are navigation links: Gestión de Cuentas, Gestión Vehículos, Gestión Reservas, Facturación, and a user profile icon for "Lucia.PatonCuenca42". Below the header, a section titled "Usuarios con Reservas" lists four users with their names and reservation details:

User	Name	Action
Yolanda	Ayuso Agúndez	Search icon
Alejandro	Sanchez Arcos	Search icon
Julian	Román Alberca	Search icon
Alejandro	Sánchez Arcos	Search icon

Como podemos observar aparecen aquellos usuarios que tienen reservas. Si pulsamos en consultar reservas (lupa) de Alejandro Sánchez Arcos:

The screenshot shows a user interface titled "Mueve TIC". At the top, there are navigation links: Gestión de Cuentas, Gestión Vehículos, Gestión Reservas, Facturación, and a user profile icon for "Lucia.PatonCuenca42". Below the header, a section titled "Historial Reservas Alejandro" lists four reservations made by Alejandro Sanchez Arcos, each with a star rating and a search icon:

Vehicle	Date	Rating	Action
Peugeot 206	2023-12-26	★★★★★	Search icon
Peugeot 206	2023-12-26	★★★★★	Search icon
Vespino	2023-12-26	★★★★★	Search icon
Xiaomi Y5	2023-12-26	★★★☆☆	Search icon

At the bottom left, there is a "Volver" button.

Observamos que nos aparecen las reservas que ha hecho Alejandro y si pulsamos en consultar (lupa) de la segunda reserva nos aparece toda la información de la reserva:

RESERVA

Matrícula
1234SAD

Modelo
Peugeot 206

Comentario
Me ha encantado este coche!!

Estado
Histórica

Fecha
2023-12-26

Valoración
★★★★★

Volver ↺

También podrá consultar **Reservas por Vehículo**:

Si selecciona esa opción le aparecerá la siguiente interfaz donde podrá filtrar por tipo de vehículo para ver todas las valoraciones que se han hecho de un vehículo (**FUNCIONALIDAD PREMIUM**):

Reservas por Vehículo

Seleccione el tipo de vehículo que deseé consultar sus valoraciones.

Tipo de Vehículo

Coche Moto Patinete

	Honda CBR	7685THN	★★★★★	
	Yamaha R1	4436TFG	★★★★★	
	Mobylette	6390HJK	★★★★★	
	Vespolino	8978KLM	★★★★★	

Si filtramos por Motos y pulsamos en Consultar Valoraciones (lupa) de la Vespolino y le damos a ícono de comentario nos aparecerá la valoración que hizo, en este caso, Alejandro Sánchez Arcos:

Valoraciones Usuarios Vespolino

Alejandro Sánchez Arcos

Valoraciones

Fecha: 2023-12-26 Comentario: Puntuación: 4;

Aceptar

Volver ↺

Proceso Facturación Administrador:

Para poder consultar la facturación de la plataforma, el administrador deberá navegar en el menú superior central y seleccionar en la opción de **Facturación** y le aparecerá la siguiente interfaz:

The screenshot shows the 'Facturación' section of the Mueve TIC platform. At the top, there are date selection fields ('/'), a search button ('Buscar por Fecha'), and a filter for vehicle type ('Todos Vehículos'). Below this is a table of invoices:

Nombre	Usuario	Tipo	Vehículo	Fecha	Precio
Factura 1	Yolanda	Patinete	Xiaomi R1	2023-11-30	5 €
Factura 2	Yolanda	Patinete	SEGWAY A1	2023-11-30	5 €
Factura 3	Alejandro	Coche	Peugeot 206	2023-11-30	5 €
Factura 4	Alejandro	Moto	Mobyllite	2023-11-30	5 €

TOTAL FACTURADO 124€

Como podemos observar el administrador podrá filtrar la facturación por tipo de vehículo y por fecha. Si no selecciona filtro de fecha le aparecerá la facturación de todos los meses y años pero si introduce un filtro de fecha, en este caso, Diciembre de 2023 y por coche:

The screenshot shows the same 'Facturación' interface as before, but with a modal dialog box overlaid. The dialog contains an information icon, the title 'Búsqueda Facturación', and a message asking if the user is sure they want to search for invoices from December 2023. It has 'Confirmar' and 'Cancelar' buttons.

Facturación

Nombre	Usuario	Tipo	Vehículo	Fecha	Precio
Factura 11	Usuario	Coche	Audi TT	2023-12-04	6 €
Factura 12	Usuario	Coche	Peugeot 206	2023-12-05	6 €
Factura 13	Usuario	Coche	Peugeot 206	2023-12-05	6 €
Factura 14	Usuario	Coche	Audi TT	2023-12-05	6 €

TOTAL FACTURADO 60€

Proceso Configuración Administrador:

Para poder cambiar alguna variable de la aplicación, el administrador deberá posarse sobre el icono de la esquina superior derecha y seleccionar la opción de **Configuración**, donde le aparecerá la siguiente interfaz:

CONFIGURACION

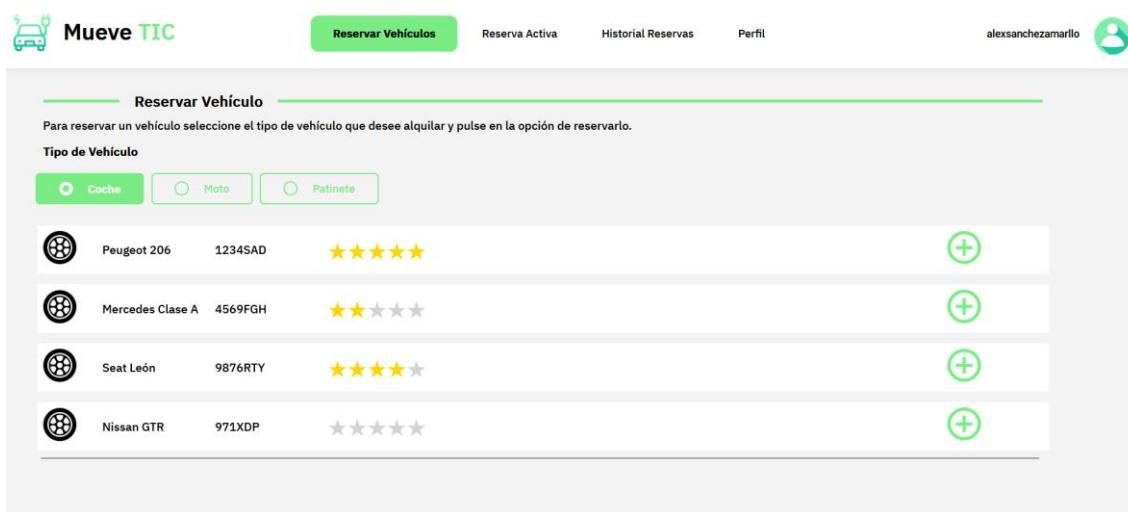
Máx.Vehículos Recargables x Personal 5	Batería x Viaje 20
Facturación x Viajes 6	Porcentaje de Batería Mínimo 20

Guarda **Volver**

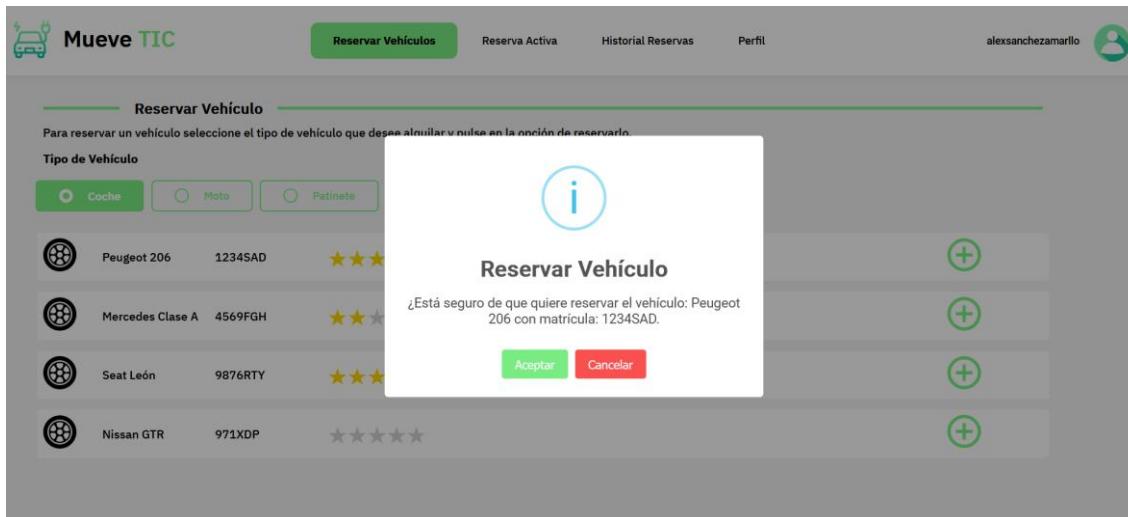
ROL USUARIO

Proceso Reservar Vehículo Usuario:

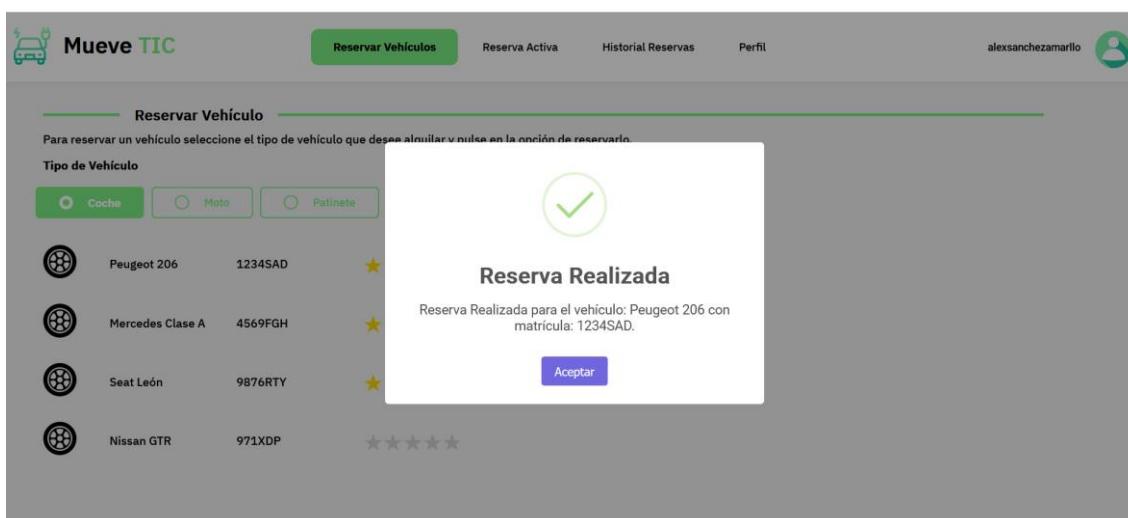
Para poder reservar un vehículo el usuario deberá navegar en el menú superior central y seleccionar la opción de **Reservar Vehículo**:



Como podemos observar en esta interfaz, el usuario podrá filtrar por el tipo de vehículo que desee alquilar pudiendo ver su valoración media (**FUNCIONALIDAD PREMIUM**). Para reservar pulsaremos sobre el ícono de añadir:



Cuando el usuario pulse en Aceptar automáticamente se creará esa reserva y se abrirá la interfaz de reserva activa.

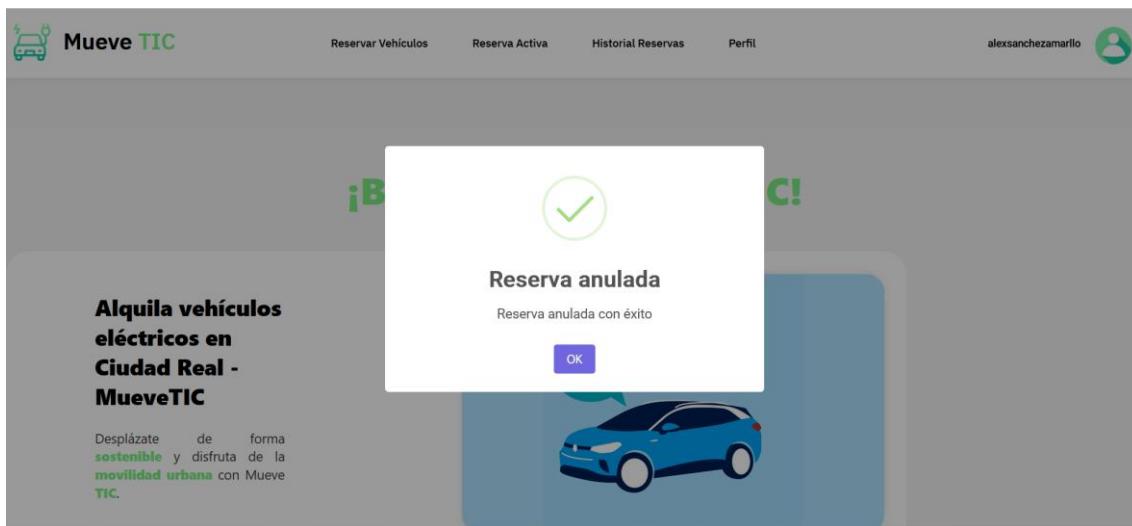


Como podemos observar los vehículos que están disponibles en la plataforma cambian de color a gris y el usuario no puede realizar otra reserva ya que solo puede hacer una a la vez.

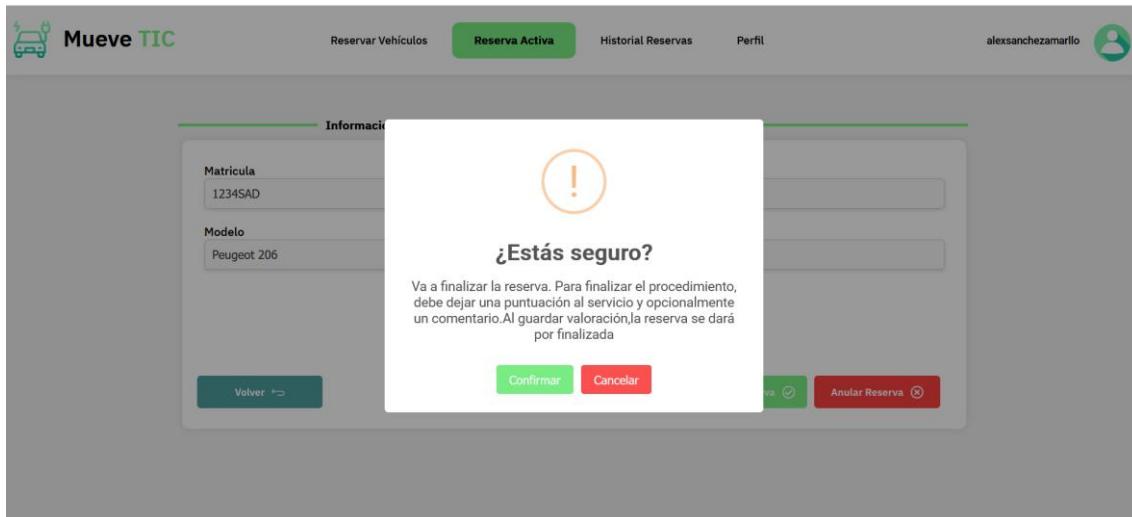
Proceso Gestionar Reserva Activa Usuario:

En esta interfaz el usuario podrá consultar toda la información de la reserva si navega en el menú superior y selecciona la opción de **Reserva Activa** pudiendo Finalizarla o Anularla.

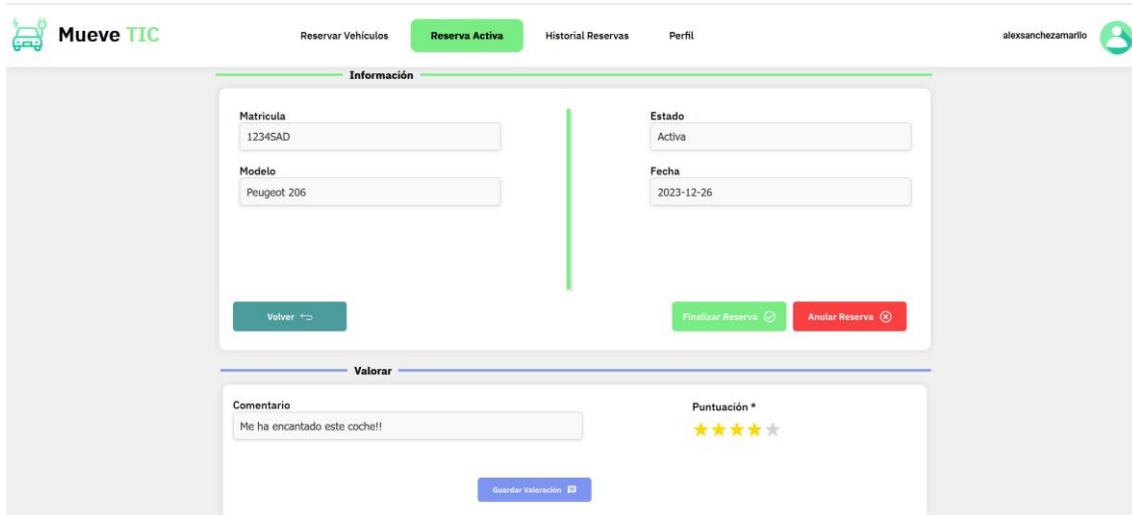
❖ Si el usuario Anula la Reserva:



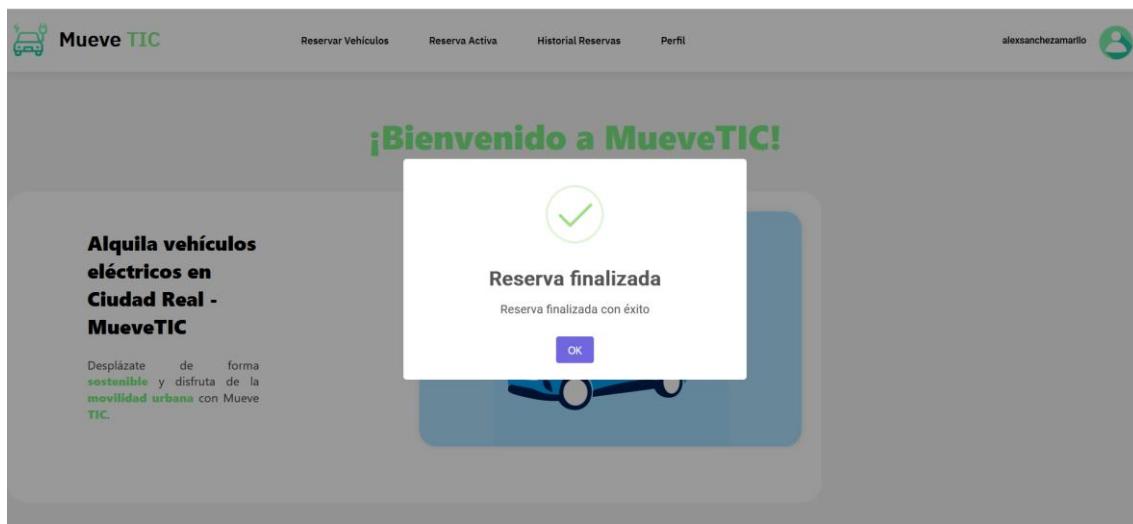
- ❖ Si el usuario la Finaliza:



Cuando el usuario seleccione confirmar aparecerá un recuadro donde podrá dejar un comentario (opcional) y una puntuación con estrellas (obligatorio) según le haya parecido el coche.



Para guardar la valoración pulsaremos en el botón y el usuario habrá finalizado la reserva:



Proceso Historial de Reservas Usuario:

Para poder ver el historial de reservas realizado por el usuario (**FUNCIONALIDAD PREMIUM**) navegará en el menú superior y seleccionará la opción de **Historial de Reservas**:

The screenshot shows the "Historial Reservas" section of the Mueve TIC website. At the top, there's a navigation bar with the Mueve TIC logo, menu items (Reservar Vehículos, Reserva Activa, Historial Reservas, Perfil), and a user profile icon. The "Historial Reservas" tab is highlighted in green. Below the header, there's a table with four rows of reservation history:

	Peugeot 206	1234SAD	Coche	2023-12-26	Cancelada
	Peugeot 206	1234SAD	Coche	2023-12-26	Histórica
	Vespino	8978KLM	Moto	2023-12-26	Histórica
	Xiaomi Y5	8526CVB	Patinete	2023-12-26	Histórica

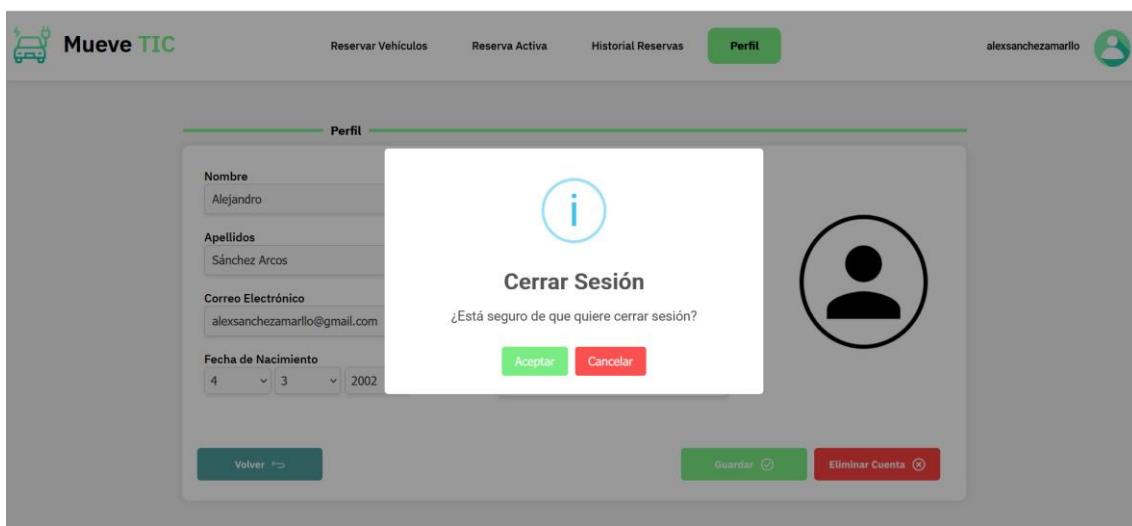
Como podemos observar en la interfaz el usuario podrá consultar la información de las reservas que hizo a lo largo del tiempo pudiendo distinguir entre las canceladas y las históricas.

Proceso Consultar Información Perfil Usuario:

Para poder consultar la información del usuario navegaremos y seleccionaremos la opción **Perfil** en el menú superior central de navegación:

Como podemos observar en la interfaz, el usuario podrá **Modificar la información de su perfil** y en el caso de que quisiera darse de baja de la plataforma podría **Eliminar su cuenta**.

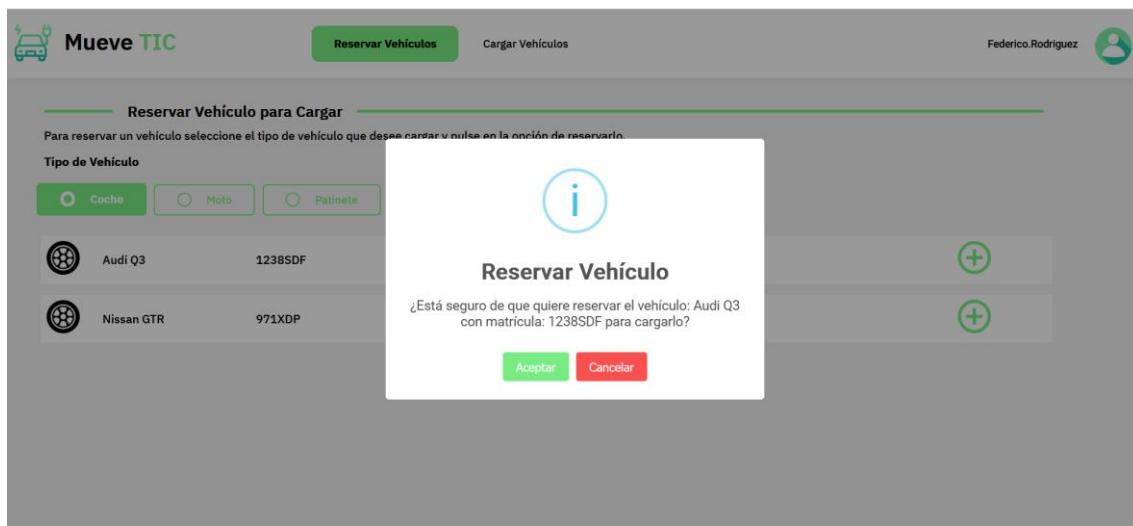
Para poder **Cerrar Sesión** (IGUAL PARA TODOS LOS TIPOS DE USUARIO) nos desplazaremos al ícono del perfil en la esquina superior derecha y pulsaremos en la opción de cerrar sesión.



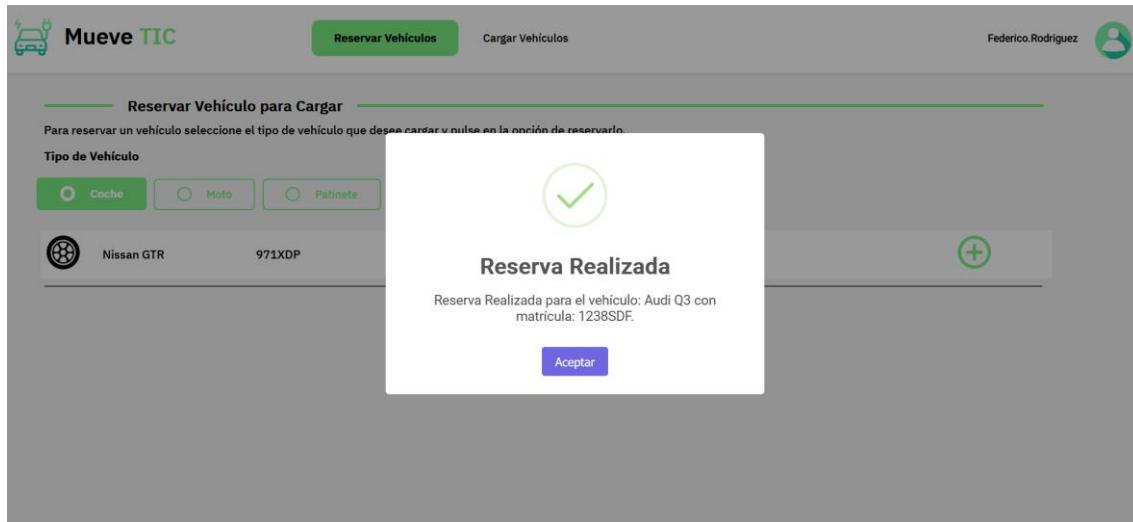
ROL PERSONAL DE MANTENIMIENTO

Proceso Reservar Vehículo para Carga Personal de Mantenimiento:

Iniciamos sesión con un personal de mantenimiento, en este caso, Federico.Rodriguez y seleccionamos en el menú superior central **Reservar Vehículo**:



Como podemos ver en la interfaz, el personal podrá hacer la reserva para cargar en función de los tipos que hay en la plataforma. Una vez pulsemos en Aceptar, lo tendremos reservado para cargarlo, en este caso, como máximo la plataforma nos dejará reservar 5 vehículos:



Proceso Gestionar Reservas para Cargar Personal de Mantenimiento:

Para poder gestionar las reservas de carga realizadas, el personal deberá seleccionar la opción del menú superior central de **Cargar Vehículos**, y le aparecerá la siguiente interfaz:

Cargar Vehículos

Para cargar un vehículo pulse el icono correspondiente de cargar del vehículo que deseé. Para cancelar la reserva de la carga de un vehículo pulse la opción correspondiente a cancelar.

Vehículo	Matrícula	Tipo	Estado Carga	Cancelar
Audi Q3	1238SDF	Coche		
Nissan GTR	971XDP	Coche		
Yamaha R1	4436TFG	Moto		
Mobylette	6390HJK	Moto		
Xiaomi R1	8975JJM	Patinete		

Para **Realizar la Carga** de un vehículo le tendrá que pulsar al icono de cargar (rayo):

Cargar Vehículo

¿Está seguro de que quiere cargar el vehículo: Audi Q3 con matrícula: 1238SDF.

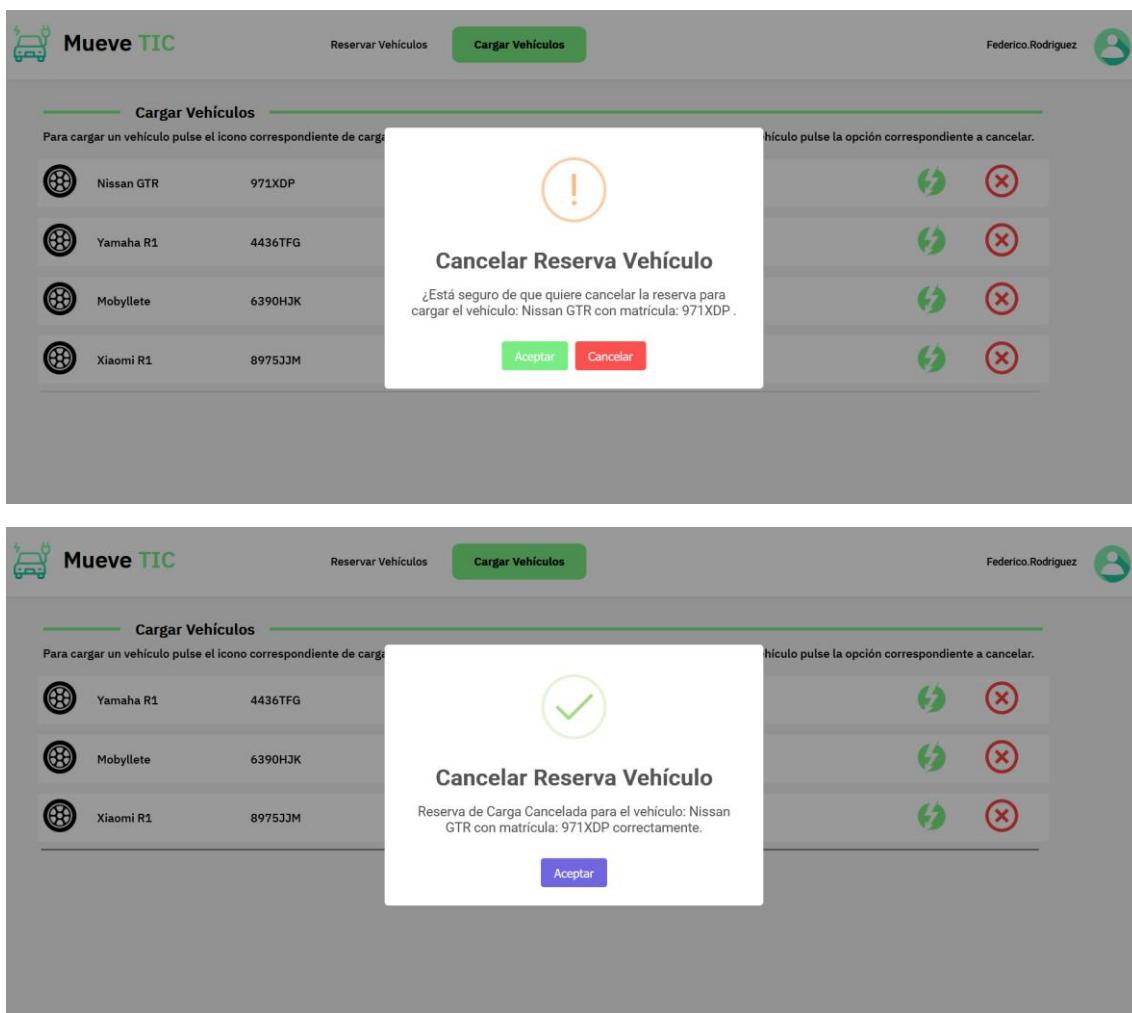
Aceptar **Cancelar**

Cargar Vehículo

Carga Realizada para el vehículo: Audi Q3 con matrícula: 1238SDF correctamente.

Aceptar

Si quisiera **Cancelar la reserva de carga** de un vehículo pulsaría sobre el icono de cancelar:



6 Mantenimiento

6.1 Análisis Crítico del Proyecto Heredado

Para realizar un análisis crítico del proyecto heredado, hemos considerado los siguientes aspectos donde hay defectos:

1) Análisis de Funcionalidades

Para realizar un análisis de las funcionalidades del proyecto hemos ejecutado el código heredado, encontrando problemas a la hora de llevar a cabo de forma correcta las siguientes funcionalidades:

- Realizar Reserva Usuario
- Cancelar Reserva Usuario
- Recuperar Contraseña

Al reducir el alcance, el grupo anterior no consideró añadir las siguientes funcionalidades:

- Añadir Valoración y Comentario al Finalizar una Reserva por parte del Usuario.
- Consultar la Facturación de la Plataforma para el Administrador.
- Funcionalidades Premium.

2) Análisis de Seguridad

En cuanto a la seguridad, el proyecto no tenía demasiados mecanismos para hacer el proyecto más seguro. Las únicas medidas de seguridad son:

- Token de sesión para los usuarios
- Encriptación de contraseñas en la base de datos con algoritmo sha512.
- Doble factor de autenticación con Google Authenticator.

Para hacer el proyecto más seguro tendrían que haber tenido en cuenta las siguientes consideraciones:

- **Parte Backend:**

- o Realizar un **Control de Peticiones Seguras** con Spring Security ya que, como podemos observar en la siguiente captura, aceptan cualquier tipo de petición de cualquier origen sin distinción de roles:

```
public class SecurityConfig{
    @Bean
    protected SecurityFilterChain securityFilter(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeHttpRequests()
            .requestMatchers("/**")
            .permitAll();
        DefaultSecurityFilterChain filterChain = http.build();
        return filterChain;
    }
}
```

Consideramos que la forma más efectiva de solucionar esta vulnerabilidad es haciendo uso de Json Web Token (JWT) para que cada usuario recibiera un token con un rol asociado para acceder a los recursos dependiendo de su rol.

- o Ausencia de **Verificación de Correo Electrónico** al registrarse en la plataforma. La forma de solucionar esta vulnerabilidad es no permitir el acceso a la plataforma si el usuario no ha verificado su correo electrónico implementando una verificación por correo.
 - o Ausencia de **Bloqueo de Cuenta** tras un número de intentos fallidos para acceder a la plataforma. Solucionaríamos esta vulnerabilidad llevando un registro del número de intentos para acceder a la plataforma, donde solo un administrador pudiera desbloquear ese usuario.
 - o Implementación de **Recuperación de Contraseña Fallida**, ya que la plataforma daba errores al intentar cambiar la contraseña de un usuario.

- **Parte Frontend:**

- o Ausencia de **Guards** para controlar el acceso a los distintos recursos de la página web dependiendo del rol y privilegios del usuario, ya que cualquier usuario podría acceder a recursos que no debería de ver. La forma de solucionar esta vulnerabilidad es implementar guards para cada tipo de usuario, pudiendo cada uno de ellos acceder a los recursos que deberían de tener asignados.

```

const routes: Routes = [
  { path: "StartingPage", component: StartingPageComponent },
  { path: "", component: StartingPageComponent },
  { path: "Register", component: RegisterComponent },
  { path: "Login", component: LoginComponent },
  { path: "Recover-Pwd", component: RecoverPwdComponent },
  { path: "VehicleList", component: VehicleListComponent },
  { path: "VehicleCard", component: VehicleCardComponent },
  { path: "Userlist", component: UserListComponent },
  { path: "EnrollVehicle", component: EnrollVehicleComponent },
  { path: "EnrollUser", component: EnrollUserComponent },
  { path: "HomePage", component: HomePageComponent },
  { path: "ReserveInfo", component: ReserveInfoComponent },
  { path: "ModifyUser", component: ModifyUserComponent },
  { path: "ModifyVehicle", component: ModifyVehicleComponent },
  { path: "ModifyReserve", component: ModifyReserveComponent },
  { path: "ToChargeList", component: VehiclesToChargeListComponent },
  { path: "ReservedToChargelist", component: ListReservedToChargeComponent },
  { path: "Consult-Info-Vehicle", component: ConsultInfoVehicleComponent },
  { path: "RecoverPwd", component: RecoverPwdComponent },

  { path: "ChangePwd", component: ChangePwdComponent },
  {
    path: "AdminBoard",
    component: AdminBoardComponent
  },
  { path: "ReserveList", component: ReserveInfoComponent }
];

```

6.2 Alcance Definido

Durante la reunión de planificación, se acordó con el Product Owner el alcance del Sprint de Mantenimiento donde se consideraron añadir las siguientes funcionalidades a la Pila del Sprint:

- Correcciones y Adición:
 - o Corregir funcionalidad Realizar Reserva Usuario.
 - o Corregir funcionalidad Recuperar Contraseña Usuario.
 - o Corregir funcionalidad Cancelar Reserva Usuario.
 - o Añadir funcionalidad de Añadir Valoración y Comentario a la Reserva Usuario
 - o Añadir funcionalidades del Rol de Gestión Telefónica.
 - o Aplicación Web Responsive y móvil.
- Mejoras:
 - o Mejorar visualmente la interfaz de las partes desarrolladas durante el Sprint de Mantenimiento.
 - o Mantener el nivel de calidad.

Respecto a la Seguridad del código heredado, se acordó durante la reunión que no se añadirían mejoras de seguridad ya que habría que hacer cambios en la base y el diseño del código, quedándose la Pila del Sprint de la siguiente forma:

1	> Corregir cancelar reserva
2	> Aplicación móvil
3	> Consultar estado de reservas (atención telefónica)
4	> Consultar estado de vehículos (atención telefónica)
5	> Eliminar reservas (atención telefónica)
6	> Modificar reservas (atención telefónica)
7	> Crear reservas (atención telefónica)
8	> Añadir rol de atención telefónica
9	> Valoración de reservas
10	> Arreglar hacer reservas
11	> Renovación de contraseñas

6.3 Addenda a la arquitectura de desarrollo

Para completar las nuevas funcionalidades acordadas durante la planificación del Sprint de Mantenimiento, analizaremos el código añadido para la parte del Backend y del Frontend:

- **Backend:**

- o **Corrección funcionalidad de Reserva de Usuario:**
Para conseguir que funcionara correctamente esta funcionalidad modificamos el código, ya que tenían errores de concordancia entre métodos con el nombre del atributo LicensePlate.
- o **Corrección funcionalidad de Recuperar Contraseña Usuario:**
Para conseguir que funcionara correctamente esta funcionalidad modificamos un parámetro en la url de sendinblue.
- o **Corrección funcionalidad de Cancelar Reserva Usuario:**
Para conseguir que funcionara correctamente esta funcionalidad modificamos el código, ya que tenían errores de concordancia entre métodos con el nombre del atributo LicensePlate.
- o **Añadir funcionalidad de Añadir Valoración y Comentario a la Reserva Usuario:**

Para añadir esta funcionalidad añadimos a la entity de reserva los atributos de valoración y comentario:

```
public Reserve(String email, String licencePlate, Date date) {
    this.email = email;
    this.licencePlate = licencePlate;
    this.date = date;
    this.state = ReserveState.ACTIVE;
    this.comment = "";
    this.punctuation = 0;
}
```

Añadimos la petición de añadir valoración al service y el controller de reserve: Controller:

```
@PutMapping("/addValoration")
public void addValoration(@RequestBody Map<String, Object> data) throws NullPointerException, JSONException {
    JSONObject vehicleData;
    Reserve reserve = null;

    try {
        userService.checkToken(data.get("token").toString(), Client.class);
        reserve = reserveService.getReserveFinish(data.get("id").toString());
        reserveService.addValoration(reserve, data.get("comment").toString(), data.get("puntuacion"));
    } catch (ClassCastException | NumberFormatException cce) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid parameter format");
    } catch (ForbiddenAccessException fae) {
        throw new ResponseStatusException(HttpStatus.FORBIDDEN, fae.getMessage());
    }
}
```

Service:

```
public void addValoration(Reserve reserve, String comment, String punctuation) {
    int punctuationInt = Integer.parseInt(punctuation);
    reserve.setPunctuation(punctuationInt);
    reserve.setComment(comment);
    reserveDAO.save(reserve);
}
```

- Añadir funcionalidades del Rol de Gestión Telefónica:

Para añadir esta funcionalidad añadimos el rol de Gestión Telefónica:

```
_id: "ayusoagundez02@gmail.com"
birthDate: 2013-12-13T00:00:00.000+00:00
drivingLicence: "A"
name: "Yoli"
surname: "Ayuso"
dni: "71369106Z"
pwd: "f5b8d8c0d1e940e71feed7b07243a42afc60333afc0a3d56f34f119659f95e6f5985e0..."
phone: "666666666"
enabled: true
type: "GESTION_TELEFONICA"
_class: "com.rentelectric.muevetic.entities.Client"
```

Añadimos las siguientes peticiones en el service y controller:

Controller:

- Para realizar la Reserva:

```
@PostMapping("/reserveVehicleGestion")
public ResponseEntity<Reserve> reserveVehicleGestion(@RequestBody Map<String, Object> data) {
    Reserve reserve = null;
    JSONObject reserveData = new JSONObject();

    if (userService.checkEmail(data.get(EMAIL).toString())) {
        try {
            String type = data.get("type").toString();
            reserveData
                .put(EMAIL, data.get(EMAIL).toString())
                .put(LICENCE_PLATE_FIELD, data.get(LICENCE_PLATE_FIELD).toString())
                .put(DATE_FIELD, data.get(DATE_FIELD).toString());

            switch (type) {
                case "USE":
                    reserve = reserveService.reserveVehicleForUseGestion(reserveData);
                    break;
                default:
                    throw new InvalidFormatException("Unrecognized type field value");
            }
        } catch (NullPointerException ne) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Missing required values");
        } catch (ClassCastException cce) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid parameter format");
        } catch (InvalidFormatException ife) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, ife.getMessage());
        } catch (IllegalArgumentException iae) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Date must be yyyy-mm-dd");
        } catch (ForbiddenAccessException fae) {
            throw new ResponseStatusException(HttpStatus.FORBIDDEN, fae.getMessage());
        } catch (EntityNotFoundException enfe) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, enfe.getMessage());
        } catch (ReserveConflictException rce) {
            throw new ResponseStatusException(HttpStatus.CONFLICT, rce.getMessage());
        }
    }
    return new ResponseEntity<>(reserve, HttpStatus.CREATED);
}
```

- Para cancelar la Reserva:

```

    @PostMapping("/cancelGestion")
    public void cancelGestion(@RequestBody Map<String, Object> data) {
        /*
         * JSON fields: token: String, used to get the user owner of the reserve
         * licencePlate: String, plate format, used to identify vehicle in maintenance
         * reserves, not checked for client reserves
         */
        try {
            JSONObject js = new JSONObject().put(EMAIL, data.get(EMAIL).toString())
                .put(LICENCE_PLATE_FIELD, data.get(LICENCE_PLATE_FIELD).toString());
            reserveService.cancelReserveGestion(js);
        } catch (NullPointerException npe) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, npe.getMessage());
        } catch (ArrayIndexOutOfBoundsException aioobe) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, "Vehicle not found");
        }
    }

    ■ Para finalizar la Reserva:
    @PutMapping("/finishGestion")
    public void finishReserveGestion(@RequestBody Map<String, Object> data) {
        try {
            this.reserveService.finishGestion(data.get(EMAIL).toString());
        } catch (NullPointerException ne) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Missing required values");
        } catch (InvalidFormatException ife) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, ife.getMessage());
        } catch (ForbiddenAccessException fae) {
            throw new ResponseStatusException(HttpStatus.FORBIDDEN, fae.getMessage());
        } catch (EntityNotFoundException enfe) {
            throw new ResponseStatusException(HttpStatus.NOT_FOUND, enfe.getMessage());
        } catch (ReserveConflictException rce) {
            throw new ResponseStatusException(HttpStatus.CONFLICT, rce.getMessage());
        }
    }

    ■ Para obtener el historial de Reservas:
    @GetMapping("/listGestionTel")
    public List<Reserve> listGestionTel(@RequestParam String token, @RequestParam String vehicleType,
                                         @RequestParam String reserveState) []
        try {
            return reserveService.list(vehicleType, reserveState);
        } catch (InvalidFormatException ife) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST, ife.getMessage());
        }
    }
}

```

Service:

```

    ■ Para realizar la Reserva:
    public Reserve reserveVehicleForUseGestion(JSONObject reserveData) throws InvalidFormatException {
        Reserve reserve = null;
        String email = reserveData.getString("email");
        String licencePlate = reserveData.getString(LICENCE_PLATE);

        checkDate(reserveData);

        Vehicle vehicle = checkVehicleExistance(licencePlate);
        if (vehicle.getState() != VehicleState.AVAILABLE) {
            throw new ReserveConflictException(VEHICLE_NOT_AVAILABLE);
        }

        if (this.reserveManager.getClientActiveReserve(email) != null) {
            throw new ReserveConflictException("User already have an active reserve");
        }

        reserve = reserveGestion(ReserveManager.USE_RESERVE, email, vehicle, reserveData.getString("date"));
        return reserve;
    }
}

```

```
private Reserve reserveGestion(String type, String email, Vehicle vehicle, String date) {
    Reserve reserve;

    // Reserve vehicle
    vehicle.setState(VehicleState.RESERVED);
    this.vehicleService.saveVehicle(vehicle);

    // Add reserve to active ones
    reserve = new Reserve(email, vehicle.getLicencePlate(), Date.valueOf(date));
    if (type.equals(ReserveManager.USE_RESERVE)) {
        reserveManager.addReserve(reserveManager.USE_RESERVE, email, reserve);
    } else if (type.equals(ReserveManager.CHARGE_RESERVE)) {
        reserveManager.addReserve(reserveManager.CHARGE_RESERVE, email, reserve);
    }
    reserveDAO.save(reserve);
}
```

- Para cancelar la Reserva:

```
public void cancelReserveGestion(JSONObject jsa) throws
NullPointerException{
    try {
        Reserve clientReserve = reserveManager.getClientActiveReserve(jsa.getString("email"));

        reserveManager.cancelClientReserve(clientReserve);
        // Free vehicle
        freeVehicle(clientReserve.getLicencePlate());

    } catch (Exception e) {
        throw new NullPointerException("User doesn't exist");
    }
}
```

- Para finalizar la Reserva:

```
public void finishGestion(String email) throws ForbiddenAccessException, EntityNotFoundException {
    Reserve clientReserve = reserveManager.getClientActiveReserve(email);

    Vehicle vehicle = checkVehicleExistance(clientReserve.getLicencePlate());
    if (vehicle.getState() != VehicleState.RESERVED) {
        throw new ReserveConflictException("Vehicle is not reserved");
    }

    useVehicle(vehicle);

    this.reserveManager.finishReserve(reserveManager.USE_RESERVE, email, clientReserve);
    clientReserve.setState(ReserveState.FINISHED);
    this.reserveDAO.save(clientReserve);
}
```

- Para obtener el historial de Reservas:

```

public List<Reserve> list(String vehicleType, String reserveState) throws
InvalidFormatException {
    List<Reserve> list = reserveDAO.findAll();

    switch(vehicleType) {
    case "ALL":
        // Case used to more easily throw exceptions when no good type is given
        break;
    case "CAR":
        list = list.stream().filter(Reserve -> vehicleService.getVehicle(Reserve.getLicencePlate));
        break;
    case "SCOOTER":
        list = list.stream().filter(Reserve -> vehicleService.getVehicle(Reserve.getLicencePlate));
        break;
    case "MOTORBIKE":
        list = list.stream().filter(Reserve -> vehicleService.getVehicle(Reserve.getLicencePlate));
        break;
    default:
        throw new InvalidFormatException("'" + vehicleType + "' is not a valid vehicle type");
    }

    switch(reserveState) {
    case "ALL":
        // Case used to more easily throw exceptions when no good type is given
        break;
    case "ACTIVE":
        list = list.stream().filter(Reserve -> Reserve.getState().equals(ReserveState.ACTIVE));
        break;
    case "CANCELLED":
        list = list.stream().filter(Reserve -> Reserve.getState().equals(ReserveState.CANCELLED));
        break;
    case "FINISHED":
        list = list.stream().filter(Reserve -> Reserve.getState().equals(ReserveState.FINISHED));
        break;
    default:
        throw new InvalidFormatException("'" + reserveState + "' is not a valid reserve state");
    }
}

```

- **Frontend:**

- o **Corrección funcionalidad de Reserva de Usuario:**

Para conseguir que funcionara correctamente esta funcionalidad modificamos la petición en la clase de vehicle.service:

```

reserveVehicle(data: any) {
    const headers = new HttpHeaders({
        'Content-Type': 'application/json',
    });

    return this.http.post(` ${Constants.route}:${Constants.PORT}/reserve/`, data, { headers: headers });
}

```

Modificamos también el reserve del ts del componente reserve.card:

```

reservar() {
    if (this.form.value["date-input"]) {
        const formData = this.form.value;
        const reserveData = {
            type: "USE",
            token: sessionStorage.getItem("token"),
            licensePlate: this.vehicleInfo.licencePlate,
            date: formData["date-input"]
        };
        this.vehicleService.reserveVehicle(reserveData).subscribe(
            (response: any) => {
                sweet.fire({
                    icon: "success",
                    title: "Reserva realizada con éxito",
                });
                this.reserveSuccess = true;
                this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
            },
            (error) => {
                sweet.fire({
                    icon: "error",
                    title: "No se puede realizar la reserva, puede que tenga una activa",
                });
                this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
            }
        );
    } else {
        sweet.fire({
            icon: "error",
            title: "Rellene la fecha.",
        });

        this.formSubmitted = true;
    }
}

```

- **Corrección funcionalidad de Cancelar Reserva Usuario:**

Para conseguir que funcionara correctamente esta funcionalidad modificamos la petición en la clase de reserve.service:

```

cancel(licensePlate: any) {
    const token = sessionStorage.getItem("token");
    const data = {
        token,
        licensePlate,
    };
    const headers = new HttpHeaders({
        'Content-Type': 'application/json',
    });

    return this.http.post(`#${Constants.route}:${Constants.PORT}/reserve/cancel`, data, { headers: headers });
}

```

Modificamos también el cancel del ts del componente reserve.card:

```

cancel() {
  let data={
    email: this.email,
    licensePlate: this.vehicle
  }

  if(this.rol!= 'GESTION_TELEFONICA'){
    this.reserveService.cancel(this.vehicle).subscribe(
      (response: any) => {
        sweet.fire({
          icon: "success",
          title: "Reserva cancelada con éxito",
        });

        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
      },
      (error) => {
        sweet.fire({
          icon: "error",
          title: "No se puede cancelar la reserva.",
          text: error
        });
        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
      }
    );
  }else{
    this.reserveService.cancelGestion(data).subscribe(
      (response: any) => {
        sweet.fire({
          icon: "success",
          title: "Reserva cancelada con éxito",
        });

        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
      },
      (error) => {
        sweet.fire({
          icon: "error",
          title: "No se puede cancelar la reserva.",
          text: error
        });
        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
      }
    );
  }
}

```

- **Añadir funcionalidad de Añadir Valoración y Comentario a la Reserva**

Usuario:

Añadimos la petición para realizar la valoración en el reserve.service:

```

addPuntuacion(data:any){
  const headers = new HttpHeaders({
    'Content-Type': 'application/json',
  });

  return this.http.put(`${Constants.route}:${Constants.PORT}/reserve/addValoration`, data, { headers: headers });
}

```

Añadimos también a la hora de finalizar la reserva que pueda realizar la valoración en el ts del componente reserve.card:

```

    finish() {
      let data={
        email: this.email,
      }
      if(this.rol== 'GESTION TELEFONICA'){
        this.reserveService.finish().subscribe(
          (response: any) => {
            sweet.fire({
              title: 'Estás seguro?',
              text: 'Va a finalizar la reserva. Para finalizar el procedimiento, debe dejar una puntuación al servicio y opcionalmente un comentario.',
              icon: 'warning',
              showCancelButton:true,
              cancelButtonColor:'#f08080',
              cancelButtonText:'Cancelar',
              confirmButtonText:'Confirmar',
            })
            .then((result) => {
              if(result.isConfirmed){
                sweet.fire({
                  title: 'Valoración',
                  icon: 'info',
                  confirmButtonText:'Confirmar Valoración',
                  html: `
                    <label style="font-size: 14px;">Introduzca un comentario para la reserva y una puntuación:</label>
                    <input type="text" id="comentario" class="swal2-input" placeholder="Introduzca un comentario" style="width: 250px;">
                    <input style="width: 250px;" type="number" id="puntuacion" class="swal2-input" placeholder="Puntuación (0-5)" max="5" min="0">
                  `,
                  allowOutsideClick: false,
                  preConfirm: () => {
                    const puntuacionInput = document.getElementById('puntuacion') as HTMLInputElement;
                    const comentarioInput = document.getElementById('comentario') as HTMLInputElement;
                    const puntuacion = puntuacionInput.value;
                    const comentario = comentarioInput.value;
                    this.puntuacionfinal= puntuacion;
                    this.comentariofinal= comentario;
                    if (!puntuacion) {
                      sweet.showValidationMessage('Debe completar el campo de la puntuación.');
                    }
                  }
                }).then((result) => {
                  let data={
                    token: sessionStorage.getItem('token'),
                    id:this.id,
                    puntuacion:this.puntuacionfinal ,
                    comment: this.comentariofinal
                  }
                  if(result.isConfirmed){
                    this.reserveService.addPunctuation(data).subscribe(
                      (response: any) => {
                        sweet.fire({
                          icon: "success",
                          title: "Valoración finalizada con éxito",
                        });
                      },
                      (error) => {
                        sweet.fire({
                          icon: "error",
                          title: "No se puede añadir la valoración.",
                          text: error
                        });
                      }
                    );
                  }
                });
                this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
              });
            });
          (error) => {
            sweet.fire({
              icon: "error",
              title: "No se puede finalizar la reserva.",
              text: error
            });
            this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
          }
        );
      }
    }
  }
}

```

Como podemos observar hemos añadido las mejoras visuales en la interfaz con diálogos y alertas.

- **Añadir funcionalidades del Rol de Gestión Telefónica:**

- Para realizar la Reserva:
ts:

```

reservarGestion() {
  if (this.form.valid) {

    const formData = this.form.value;
    const reserveData = {
      type: "USE",
      email:formData["email"] ,
      licensePlate: this.vehicleInfo.licencePlate,
      date: formData["date-input"]
    };
    this.vehicleService.reserveVehicleGestion(reserveData).subscribe(
      (response: any) => {
        sweet.fire({
          icon: "success",
          title: "Reserva realizada con éxito",
        });
        this.reserveSuccess = true;
        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });

      },
      (error) => {
        sweet.fire({
          icon: "error",
          title: "No se puede realizar la reserva",
          text:"Puede que tenga una activa o que el usuario no sea válido"
        });
        this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
        console.log(error);
      }
    );
  } else {
    this.formSubmitted = true;
  }
}

```

Petición:

```

reserveVehicleGestion(data: any) {
  const headers = new HttpHeaders({
    'Content-Type': 'application/json',
  })
  return this.http.post(`${Constants.route}:${Constants.PORT}/reserve/reserveVehicleGestion`, data, { headers: headers });
}

```

- Para cancelar la Reserva:

ts:

```

},
} else{
  this.reserveService.cancelGestion(data).subscribe(
    (response: any) => {
      sweet.fire({
        icon: "success",
        title: "Reserva cancelada con éxito",
      });

      this.router.navigateByUrl("HomePage", { state: { type: this.rol } });

    },
    (error) => {
      sweet.fire({
        icon: "error",
        title: "No se puede cancelar la reserva.",
        text: error
      });
      this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
    }
  );
}
}

```

Petición:

```
cancelGestion(data: any) {
  const headers = new HttpHeaders({
    'Content-Type': 'application/json',
  });

  return this.http.post(`${Constants.route}:${Constants.PORT}/reserve/cancelGestion`, data, { headers: headers });
}
```

- Para finalizar la Reserva:

ts:

```
}else{

  this.reserveService.finishGestion(data).subscribe(
    (response: any) => {
      sweet.fire({
        icon: "success",
        title: "Reserva finalizada con éxito",
      });

      this.router.navigateByUrl("HomePage", { state: { type: this.rol } });

    },
    (error) => {
      sweet.fire({
        icon: "error",
        title: "No se puede finalizar la reserva.",
        text: error
      });

      this.router.navigateByUrl("HomePage", { state: { type: this.rol } });
    }
  );
}
```

Petición:

```
finishGestion(data: any) {
  const headers = new HttpHeaders({
    'Content-Type': 'application/json',
  });

  return this.http.put(`${Constants.route}:${Constants.PORT}/reserve/finishGestion`, data, { headers: headers });
}
```

- Para obtener el historial de Reservas:

ts:

```
async fetchData() {
  try {
    let data;

    switch (this.accion) {
      case 'Mi Reserva':
        data = await this.reserveService.getMyReserve();
        break;
      case 'Lista de reservas':
        data = await this.reserveService.getReserveList("ALL", "ALL");

        if(this.rol==="GESTION_TELEFONICA"){
          data = await this.reserveService.getReserveListGestion("ACTIVE", "ALL");
        }
        break;
    }

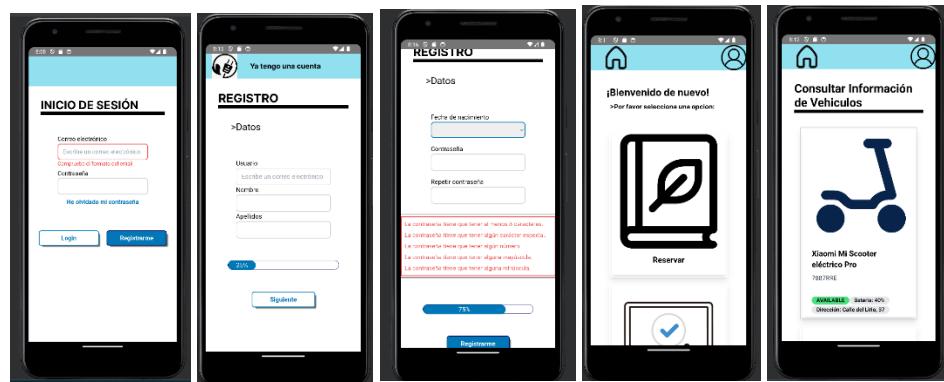
    this.reserveList = data;
    console.log(typeof this.reserveList);
    console.log(data);
    this.checkReserve();
    this.dataLoaded = true
  } catch (error) {
    console.error("Error:", error);
  }
}
```

Petición:

```
async getReserveListGestion(reserveState: string, vehicleType: string) {
    const token = sessionStorage.getItem("token");
    const url = `${Constants.route}:${Constants.PORT}${Constants.reserve}${Constants.ListGestionTel}?token=${token}&vehicleType=${vehicleType}&reserveState=${reserveState}`;
    const options = {
        method: "GET",
        headers: {
            "Content-Type": "application/json",
        },
    };
    const data = await fetch(url, options);
    return await data.json();
}
```

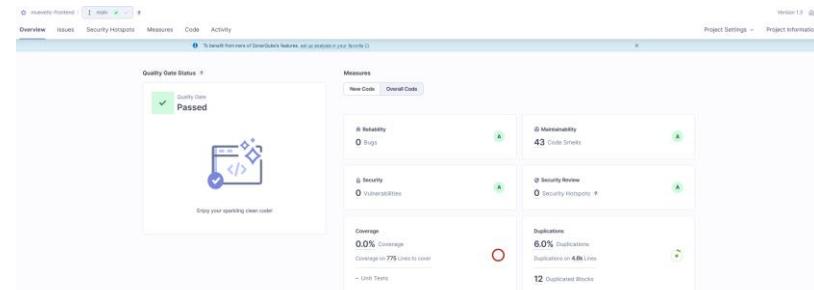
- Aspectos Generales:

- Aplicación Responsive y Móvil:



- Mantenimiento de la Calidad:

- Parte Front:



- Parte Back:

