**Asignatura: Taller de Diseño y desarrollo de Soluciones**

**Sección: D-B50-N4-P12-C1**

**Nombre del docente:** Manuel Reveco

**Nombre de los integrantes del grupo:** Esteban Astroza, Agustin Balcazar, Ayrton Bastias, Lucas Cárcamo.

**Fecha de entrega:**

**Evaluación N°4**

**Contenido**

[I. Prueba de caja blanca 3](#_Toc185012648)

[II. Preguntas 42](#_Toc185012649)

[III. Bibliografía 44](#_Toc185012650)

1. ColectiGo!

Problemática:

* Falta de visibilidad en tiempo real

Las empresas enfrentan dificultades para monitorear la ubicación de sus vehículos de manera precisa y oportuna, lo que limita la supervisión de las operaciones, aumenta los riesgos de seguridad y reduce la eficiencia operativa.

* Gestión ineficiente de conductores y vehículos

La administración manual o desorganizada de los datos de conductores y vehículos provoca errores, redundancias y pérdida de tiempo, afectando la productividad y la toma de decisiones informadas.

Objetivo General:

Desarrollar un sistema de monitoreo GPS vehicular que permita la gestión eficiente de flotas empresariales, asegurando el cumplimiento normativo mediante el rastreo en tiempo real y funcionalidades administrativas avanzadas.

Objetivos Específicos:

* Implementar un sistema de rastreo en tiempo real para monitorear la ubicación de los vehículos mediante tecnología GPS.
* Diseñar una interfaz intuitiva que permita a los administradores gestionar conductores y vehículos de manera eficiente.
* Incorporar funcionalidades para la creación, modificación y eliminación de perfiles de conductores asegurando la integridad de los datos.
* Facilitar la visualización en tiempo real de todos los vehículos en un mapa, incluyendo información relevante del conductor asociado, utilizando la tecnología Leaflet para el mapa y OwnTracks para la ubicación,
* Garantizar la seguridad de los datos mediante un sistema de autenticación exclusivo para administradores.
* Integrar servicios de terceros como Leaflet y AWS RDS para optimizar la funcionalidad y el almacenamiento del sistema.
* Diseñar la infraestructura técnica para asegurar escalabilidad, disponibilidad y conectividad continua del sistema.

1. Prueba de caja blanca
   1. **Inicio de sesión**
      1. Funcionalidad o código testeado:

Texto

Descripción generada automáticamente

* + 1. ¿Qué busca evaluar con la prueba?:

Al enviar credenciales válidas a través de un formulario POST la página redirige al menú y al enviar credenciales inválidas a través de un formulario POST no redireccione a ninguna página.

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

Credenciales Inválidas:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente  
  
Credenciales Válidas:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente   
Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* 1. **Cerrar Sesión:**
     1. Funcionalidad o código testeado:

**Texto

Descripción generada automáticamente**

* + 1. ¿Qué busca evaluar con la prueba?:

El cierre de sesión y redirección hacia el LogIn, esto estando con un usuario autenticado, se cumplirá gracias al método logout importado desde “django.contrib.auth”, esta función se le asigna una URL:  


La cual se llamará desde un “Botón” llamado “Cerrar Sesión” ubicado en el template del dashboard del usuario:

Texto

Descripción generada automáticamente

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

Resultado:

Al apretar el botón de “Cerrar sesión” la página te redirige hacia la pantalla de “Iniciar Sesión” como se esperaba.

* 1. **Visualización de los Usuarios**
     1. Funcionalidad o código testeado:

**Texto

Descripción generada automáticamente**

* + 1. ¿Qué busca evaluar con la prueba?:

La visualización de los usuarios (administradores) existentes, con el método “filter()” que filtra los resultados por los parámetros “tipo\_usuario”, lo que retorna una lista definida como “usuarios” la cual se enviará al template “gestiónUsuarios.html” en modo de diccionario clave:valor, para poder iterar desde el template con un ciclo “for”:

**Texto

Descripción generada automáticamente**

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Visualización de la lista de Usuarios:**

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media**

**Base de datos:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Resultado:**

La lista de Usuarios se muestra de forma correcta con el filtro aplicado correctamente.

**Observación:**

El tipo de usuario no debería ser “Administrador”, ya que no estamos revisando a los administradores, sino que a los usuarios que ocuparan la aplicación.

* 1. **Creación de usuarios** 
     1. Funcionalidad o código testeado:

**Texto

Descripción generada automáticamente**

* + 1. ¿Qué busca evaluar con la prueba?:

Validar la creación de un Usuario con un formulario y el método “is\_valid()”, asegurando la actualización de datos en la base de datos asignada con el método “save()”, y redirigiendo la pantalla a la gestión de usuarios.

También se probará qué sucede en el caso de que los datos sean inválidos

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

Ruta hasta la creación del Usuario

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente** **Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente**

Dato del correo Inválido:

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

Rut vacío:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Datos válidos para crear al usuario:

Texto

Descripción generada automáticamente con confianza media

Visualización:

Tabla

Descripción generada automáticamente con confianza media

Resultados:

El formulario tiene validación de datos y con alerta de campos obligatorios, en el caso de agregar datos válidos se guarda en la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* 1. **Modificar Usuario**
     1. Funcionalidad o código testeado:

**Texto

Descripción generada automáticamente**

* + 1. ¿Qué busca evaluar con la prueba?:

La capacidad del código para permitir la modificación de un usuario utilizando un formulario similar al de creación y guardando los cambios viéndose reflejado en la base de datos asignada.

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente** **Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Se visualiza el Usuario creado:**

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica

Descripción generada automáticamente**

**Acceder a la opción de “modificar”:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Formulario para modificar el usuario:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Nuevos Datos:**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Visualización del Usuario modificado:**

**Imagen que contiene Tabla

Descripción generada automáticamente**

**Datos modificados correctamente:**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**Resultado:

Se ingresaron datos para la modificación y se guardaron los cambios de forma correcta, esto se ve reflejado en la visualización de los usuarios y en la base de datos.

Interfaz de usuario gráfica, Texto, Aplicación, Word, Correo electrónico

Descripción generada automáticamente

* 1. **Eliminar Usuario** 
     1. Funcionalidad o código testeado:

**Texto

Descripción generada automáticamente**

* + 1. ¿Qué busca evaluar con la prueba?:

Completar una eliminación de un usuario buscándolo primero por el id con el método “get(id=id)” donde se le proporciona el id desde una url:



Se le extrae la id desde el template que muestra en bucle todos los usuarios:

Texto

Descripción generada automáticamente

* + 1. Responsable del Testing:

Lucas Cárcamo

* + 1. Resultado del Testing:

Para esta prueba eliminaremos un Usuario antes creado, en este caso “Test Modificar2” el cual creamos en la prueba anterior:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Visualización en la lista de usuarios:

Captura de pantalla de un celular

Descripción generada automáticamente

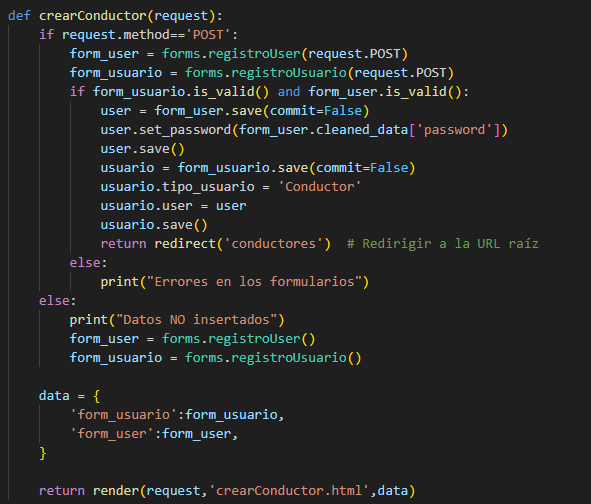
Resultado:

Al seleccionar la opción de “eliminar” después de haber seleccionado “ver detalles” en un Usuario, se elimina correctamente, y esto se ve reflejado en la base de datos:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* 1. **Crear Conductor**
     1. Funcionalidad o código testeado:

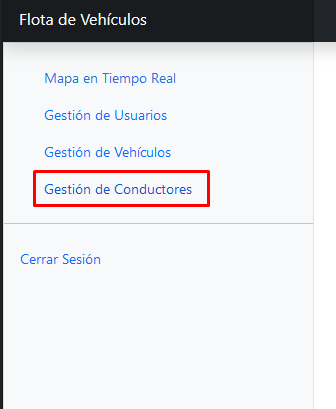


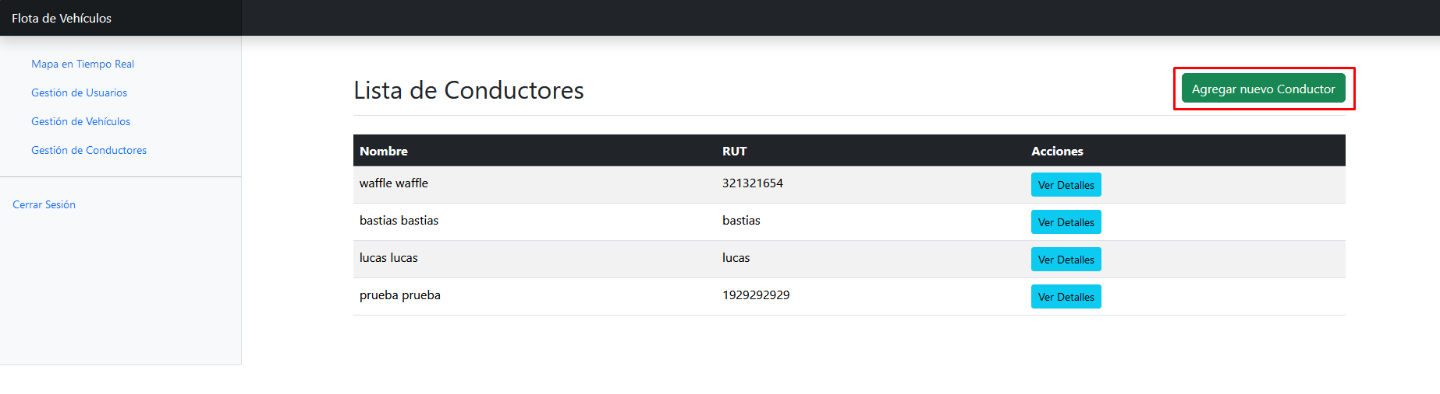
* + 1. ¿Qué busca evaluar con la prueba?:

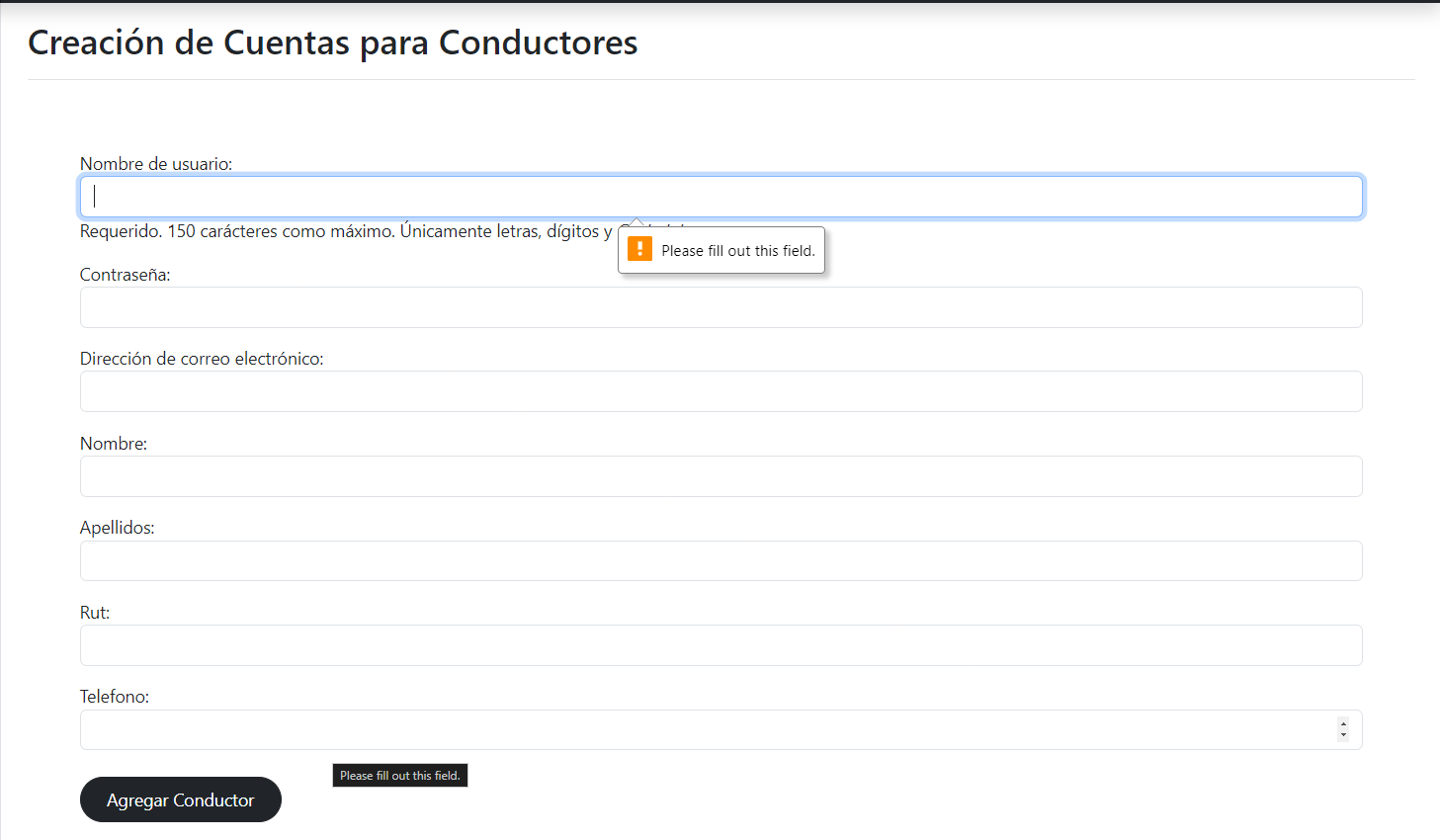
Validar la creación de un Conductor, asegurando el ingreso de los datos a la base de datos asignada con el método “save()”, y redirigiendo la pantalla a la gestión de usuarios.

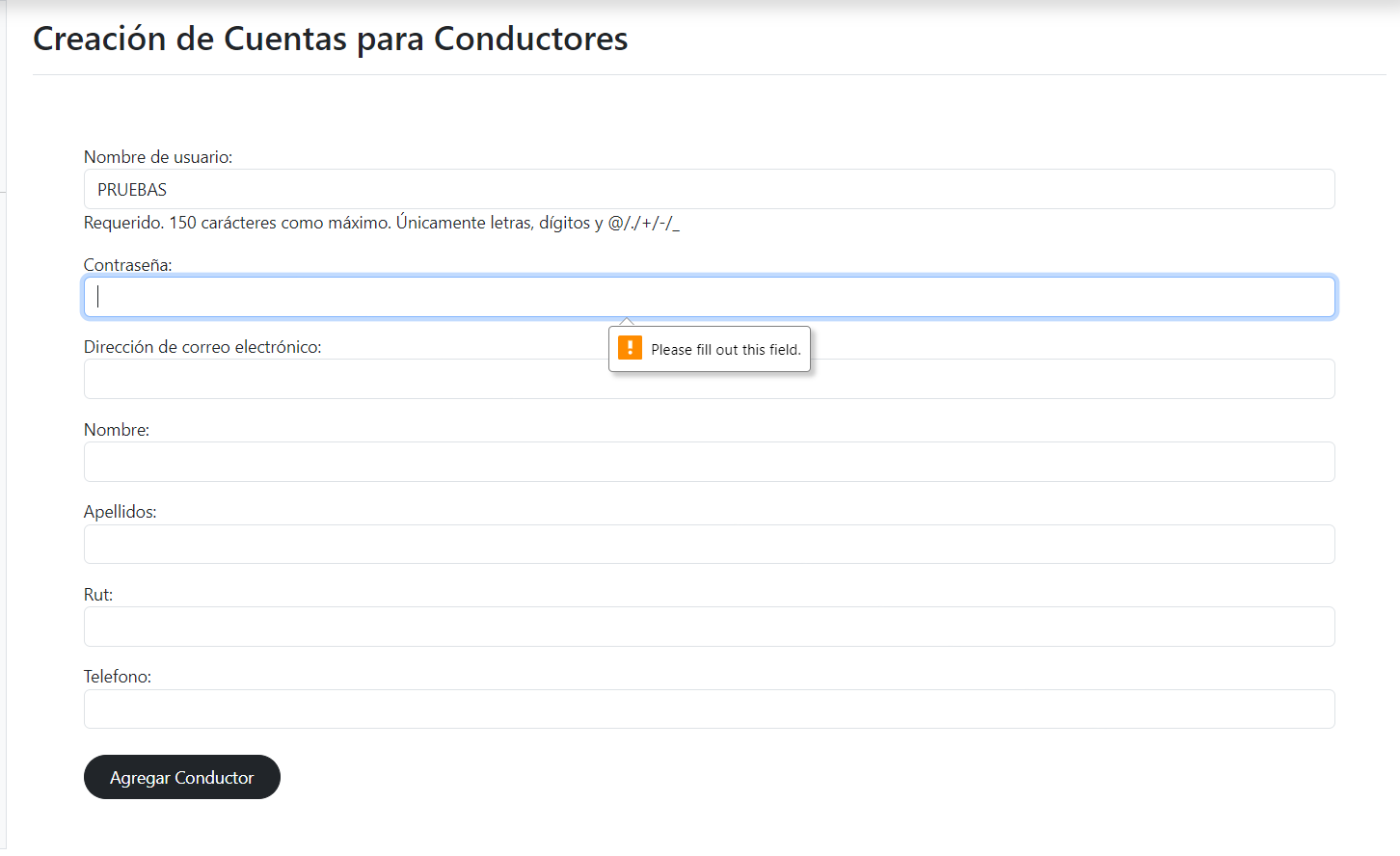
También se probará qué sucede en el caso de que los datos sean inválidos

* + 1. Responsable del Testing: Ayrton Bastias
    2. Resultado del Testing:

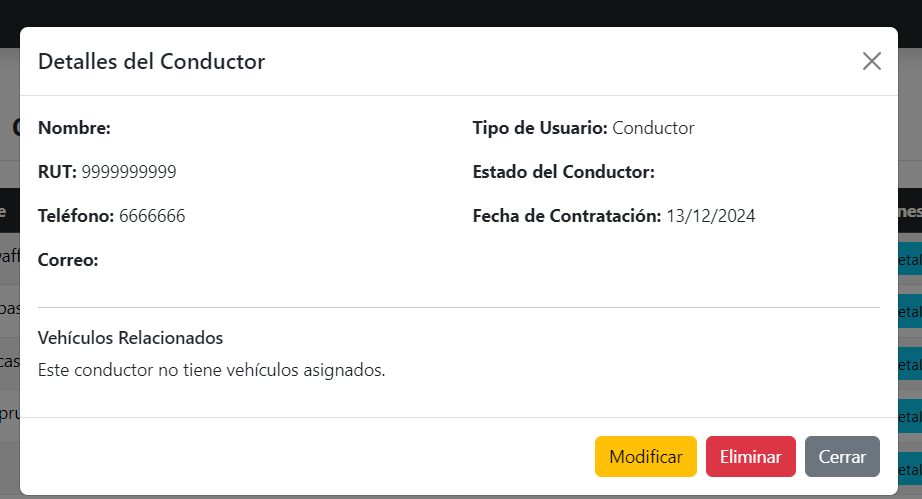


Intento de ingreso de usuario en blanco:

Intento de ingreso de Conductor sin Contraseña:

Intento de Ingreso de Conductor sin RUT:

Intento de Ingreso de Conductor sin telefono:

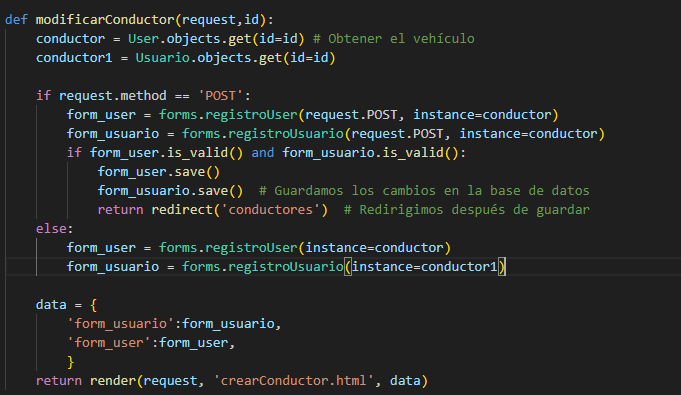
Intento de Ingreso de Conductor sin Nombre, Apellido y Correo Electronico:  
Ingresa los Datos a la base de datos.



**Observación**: Debería requerir el Nombre y Apellido del conductor antes de ingresar los datos a la BDD.

Resultado: Ingresa los datos correctamente a la BDD.

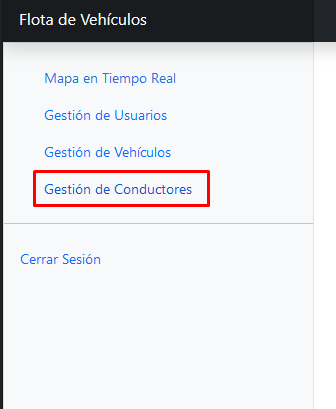
* 1. **Modificación de Conductor**
     1. Funcionalidad o código testeado:

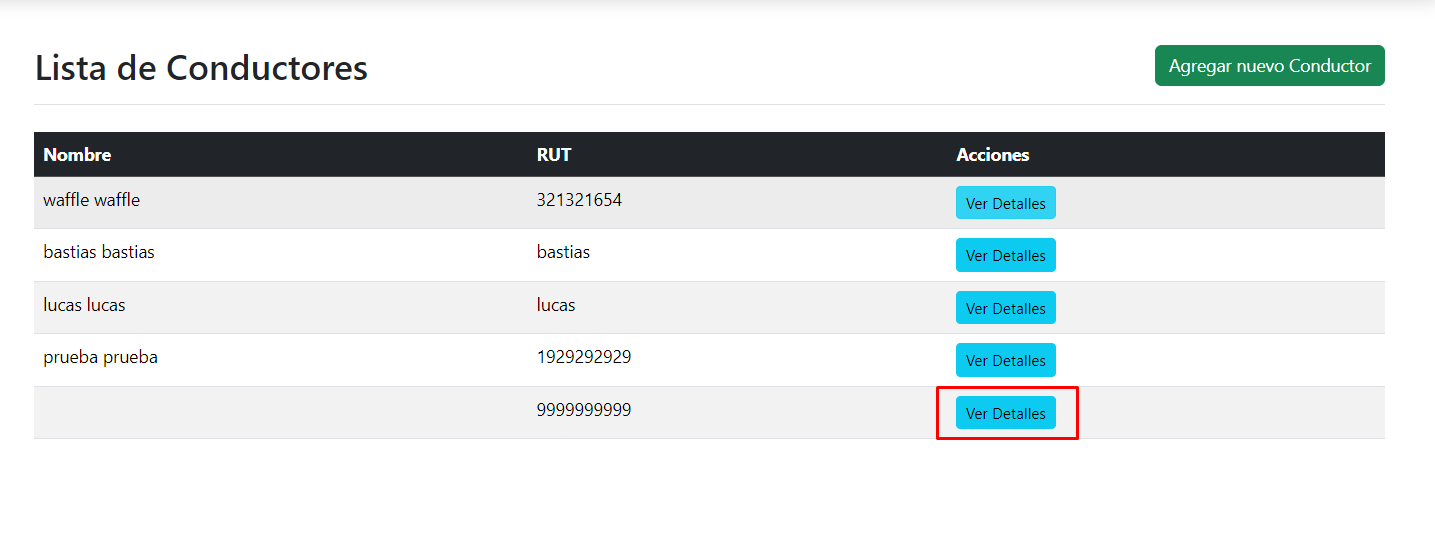


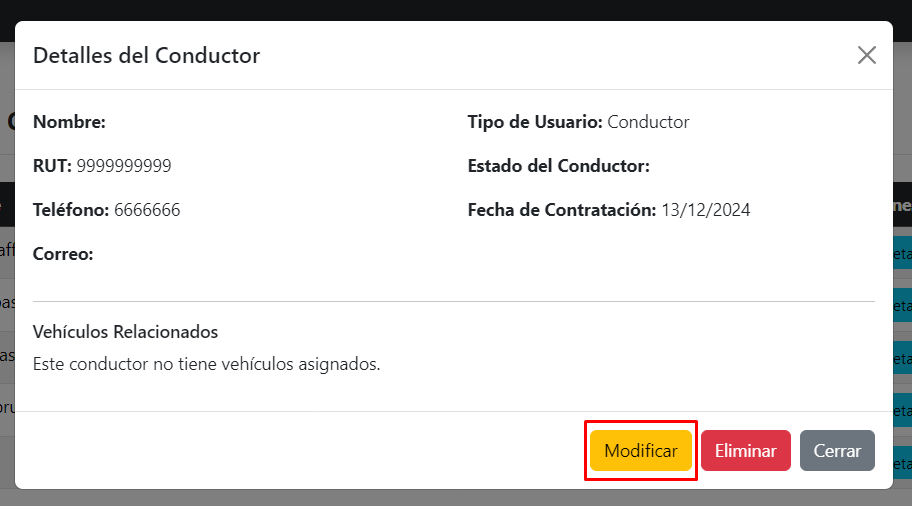
* + 1. ¿Qué busca evaluar con la prueba?:

La prueba busca evaluar la capacidad de la página web de realizar la modificación de un conductor y que los cambios realizados sean reflejados en la base de datos.

* + 1. Responsable del Testing: Ayrton Bastias
    2. Resultado del Testing:



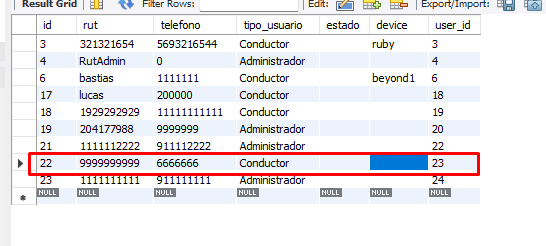
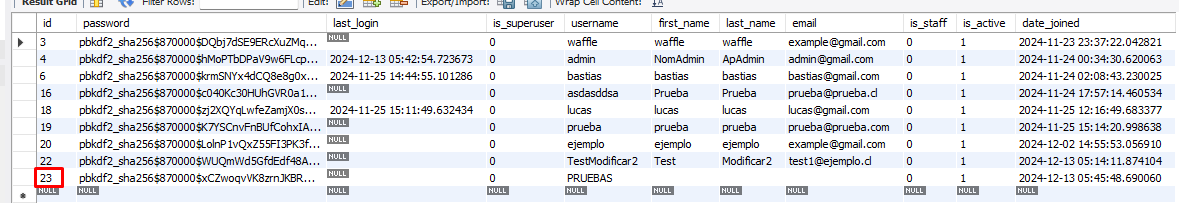




Observación: Al dar click a modificar, los datos del conductor no corresponden.



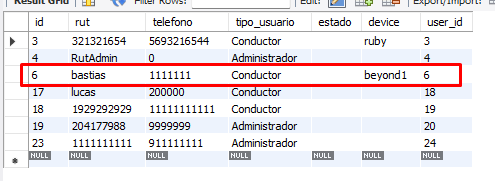
Se ve que, en la base de datos el ID del Usuario (para iniciar sesión) y el ID del conductor (Con los datos) no son iguales.



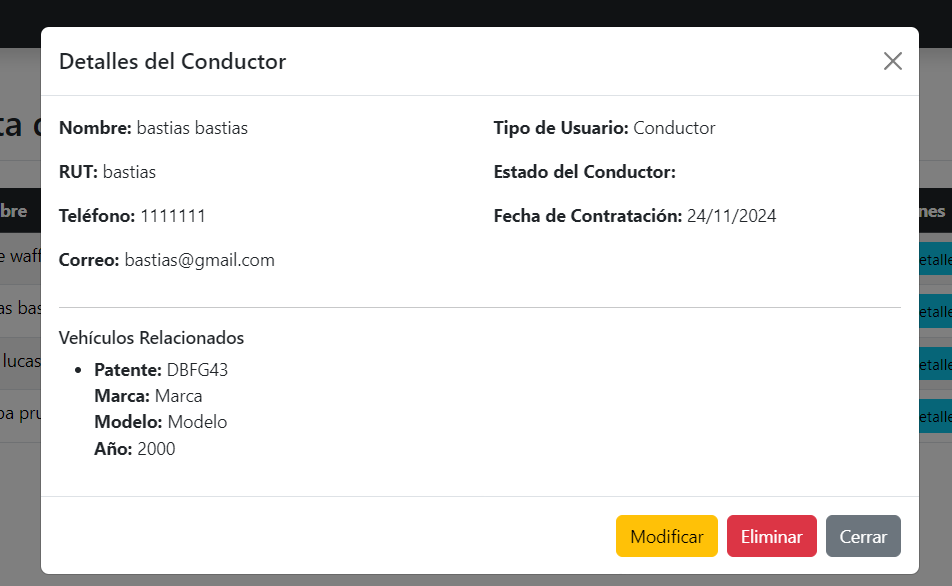
Causando que, al cargar los datos al formulario, se carguen los del Usuario con ID anterior.

Resultado: Al modificar se Modificaron los datos al usuario que no Corresponde (en este caso).

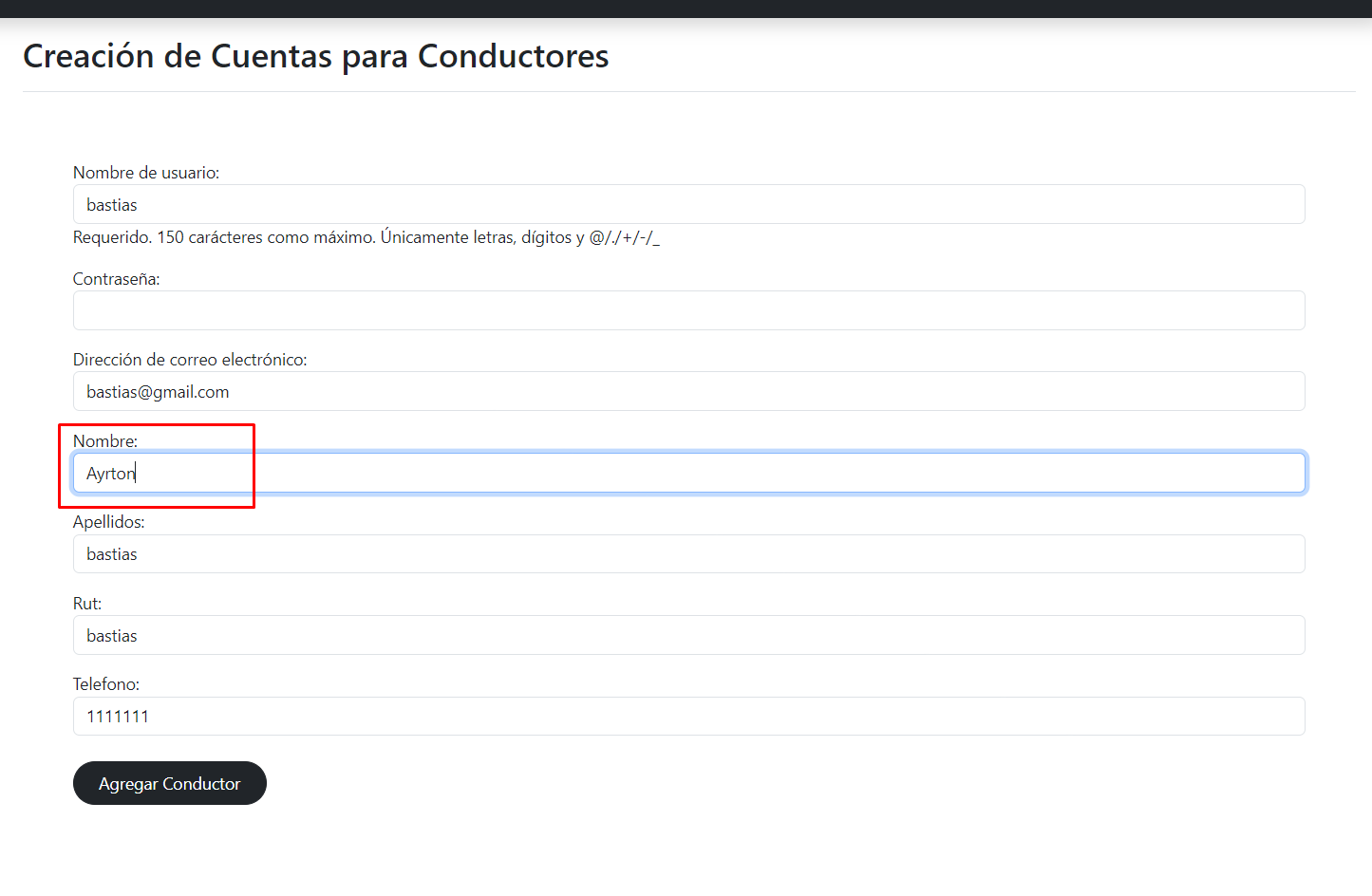
Al modificar un Usuario el cual tiene los 2 ID con el mismo valor.

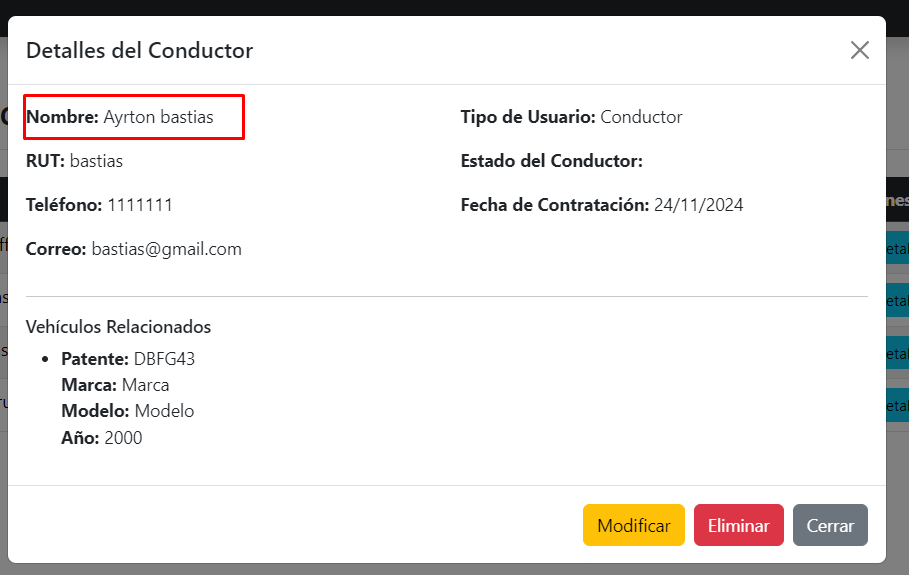


Los datos cargan correctamente y se realiza la modificación sin problemas.

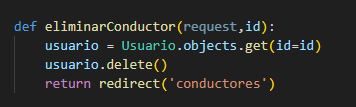


Dato a Modificar: Nombre





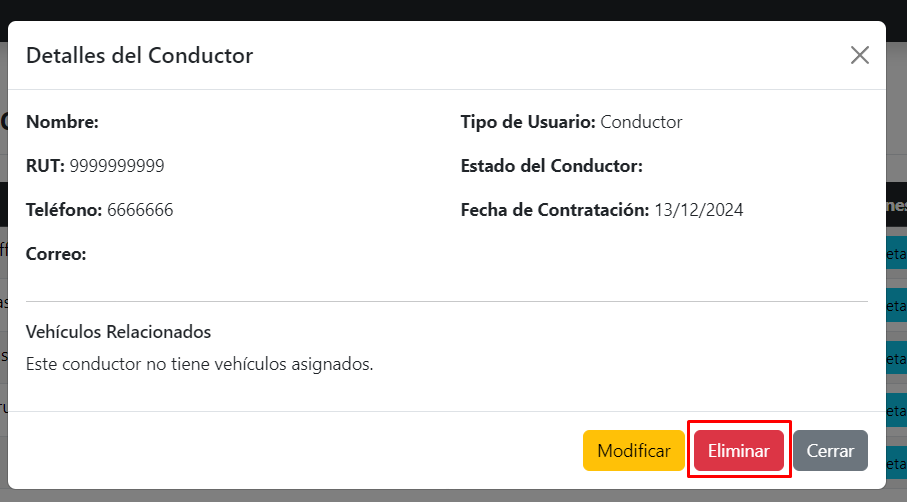
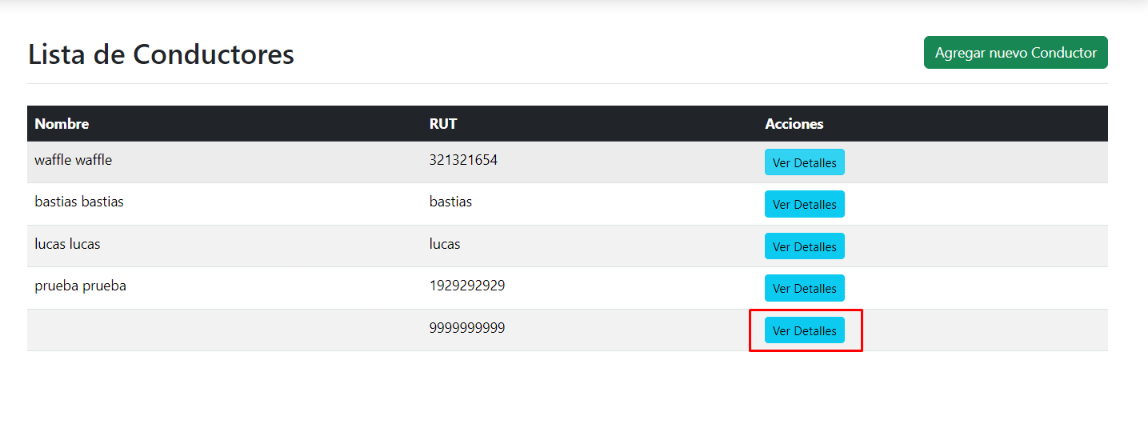
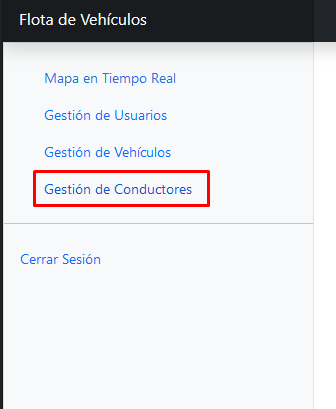
* 1. **Eliminación de Conductor**
     1. Funcionalidad o código testeado:



* + 1. ¿Qué busca evaluar con la prueba?:

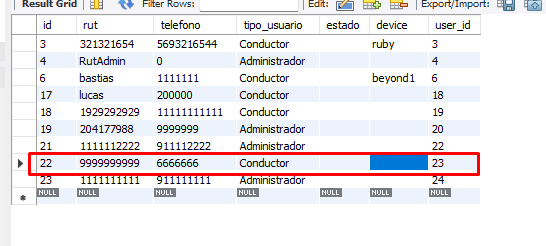
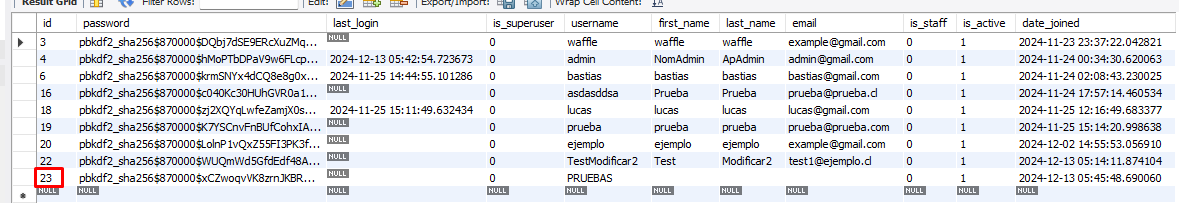
La prueba busca evaluar la capacidad de eliminar usuarios desde la página web.

* + 1. Responsable del Testing: Ayrton Bastias
    2. Resultado del Testing:



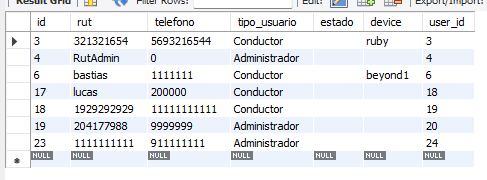
Al dar click al botón de Eliminar, El Conductor se elimina de la vista de Conductores siendo reflejado en la Base de datos.

**Antes:**

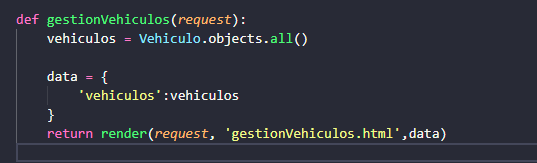


**Después:**





* 1. **Listar vehículos**
     1. Funcionalidad o código testeado:



* + 1. ¿Qué busca evaluar con la prueba?:

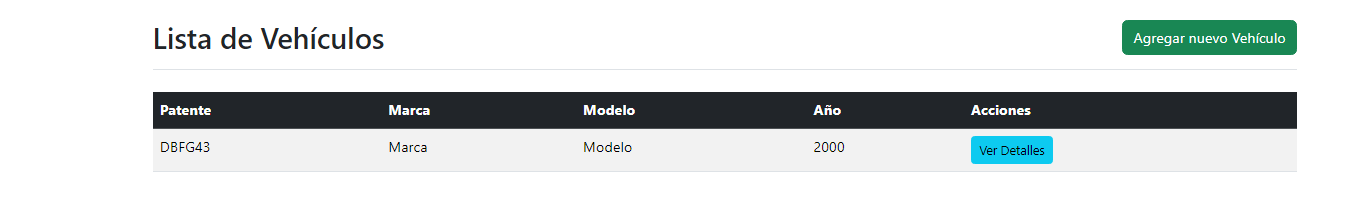
Esta prueba busca verificar el correcto funcionamiento de la función que permite listar los vehículos que ya están ingresados.

* + 1. Responsable del Testing: Esteban Astroza
    2. Resultado del testing:

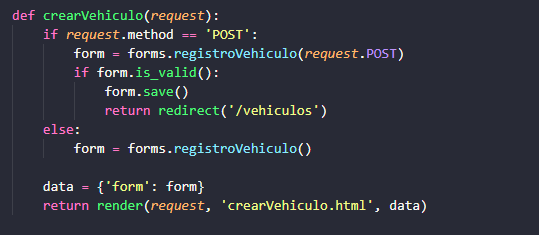
Aquí en la vista de administrador, nos vamos a donde dice “gestión de vehículos”



Para luego obtener, efectivamente, los vehículos que fueron ingresados a la base de datos:



* 1. **Crear Vehículo**
     1. Funcionalidad o código testeado:

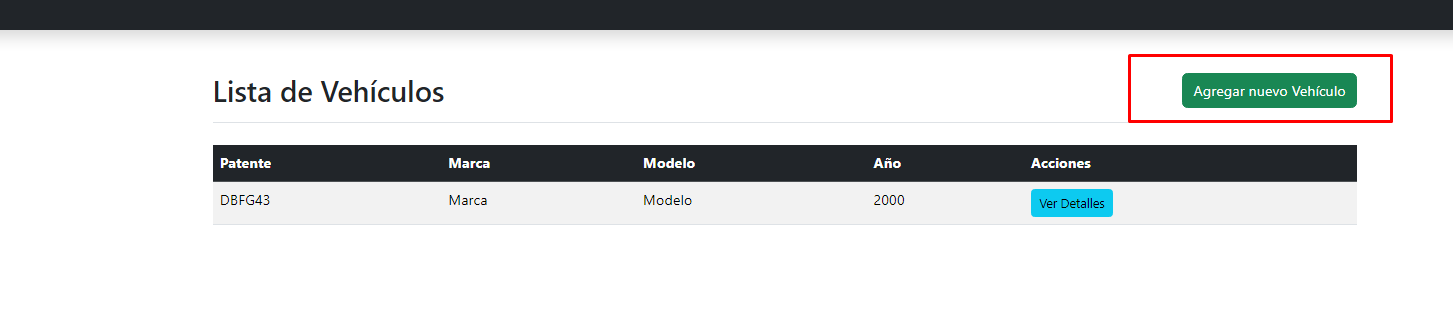


* + 1. ¿Qué busca evaluar con la prueba?:

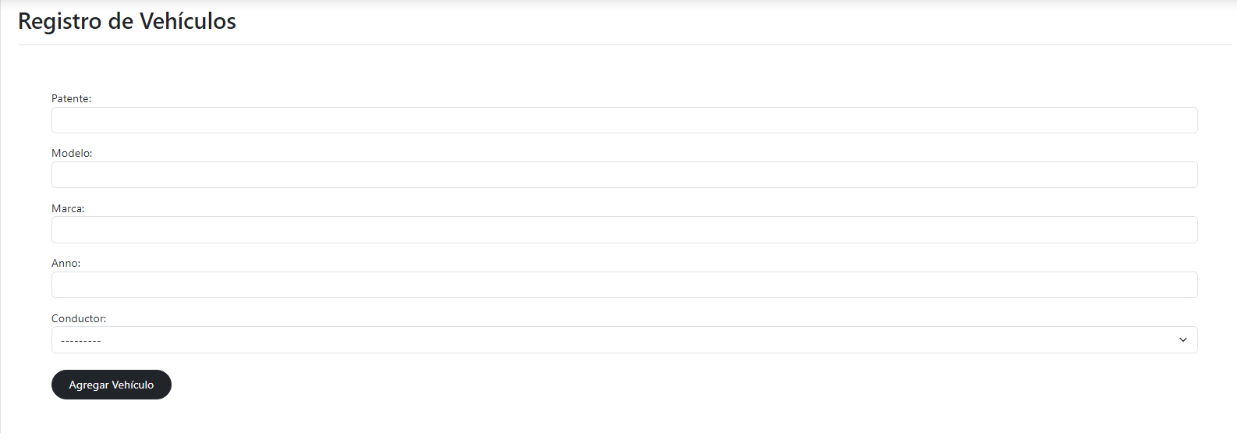
El correcto funcionamiento del formulario que permite crear vehículos.

* + 1. Responsable del Testing: Esteban Astroza
    2. Resultado del Testing:

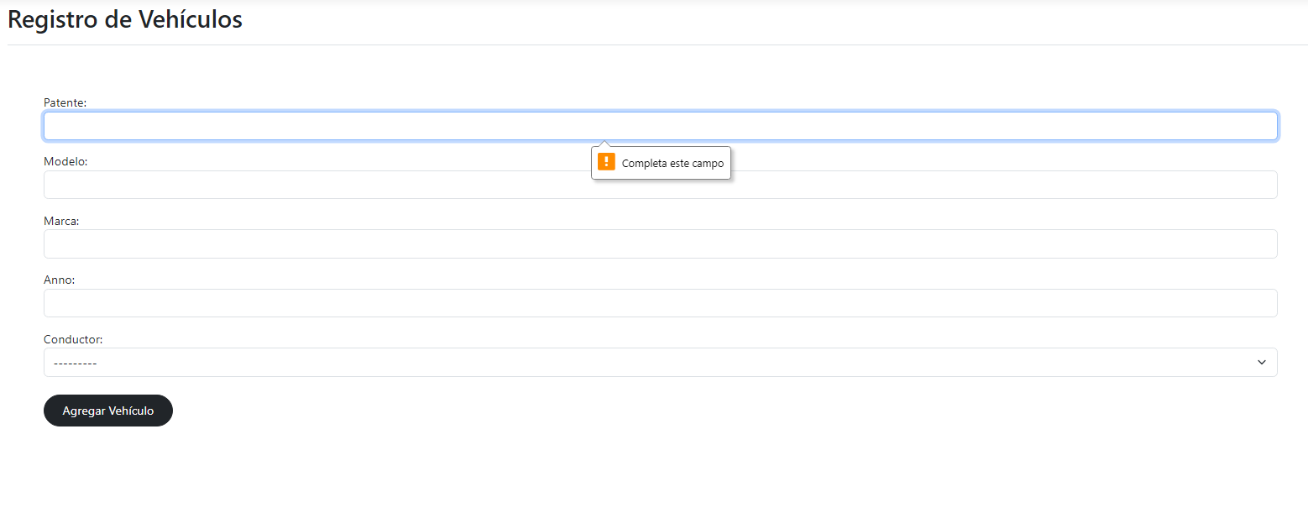
En la parte de listar vehículos, estará este botón para agregar un nuevo vehículo



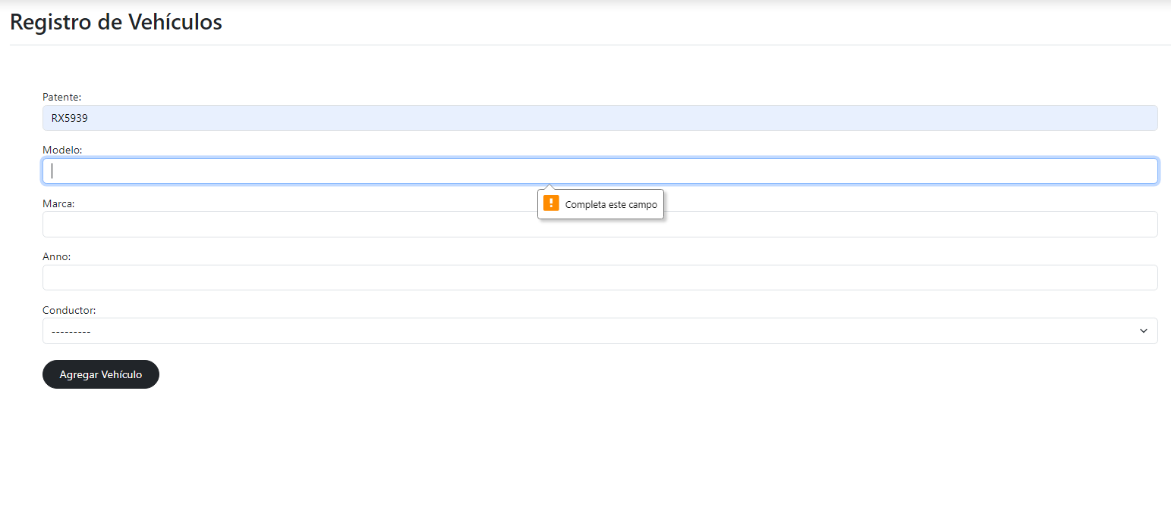
El cual redirigirá al formulario para crear vehículos.

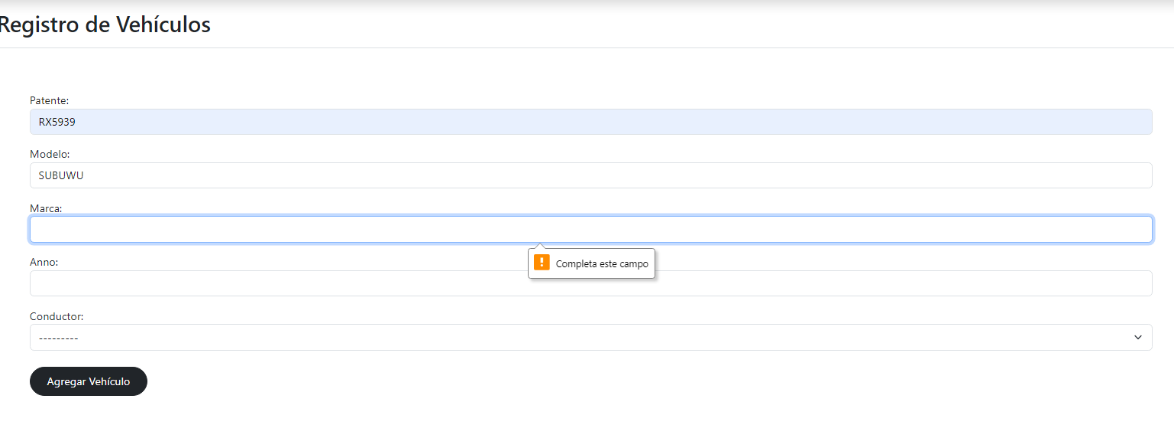


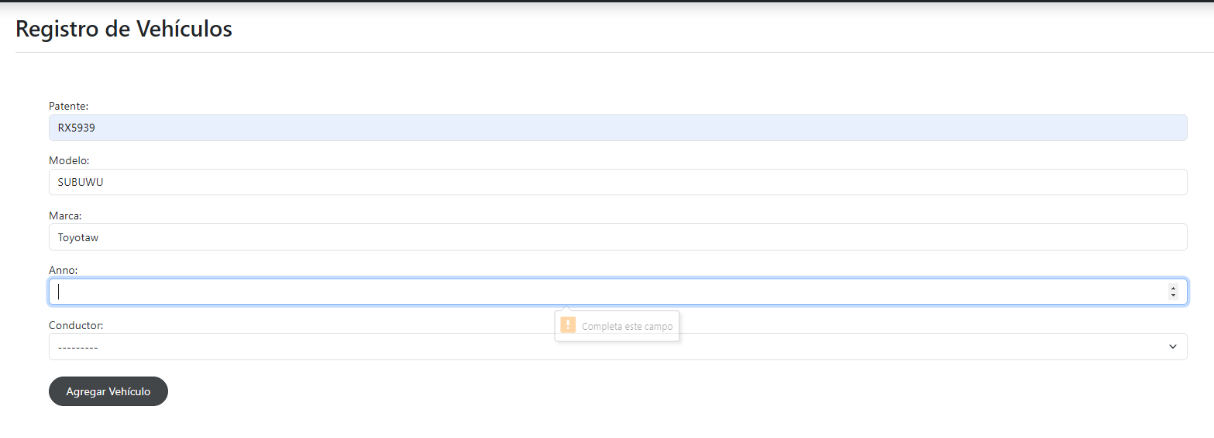
Ingresamos campos en blanco:

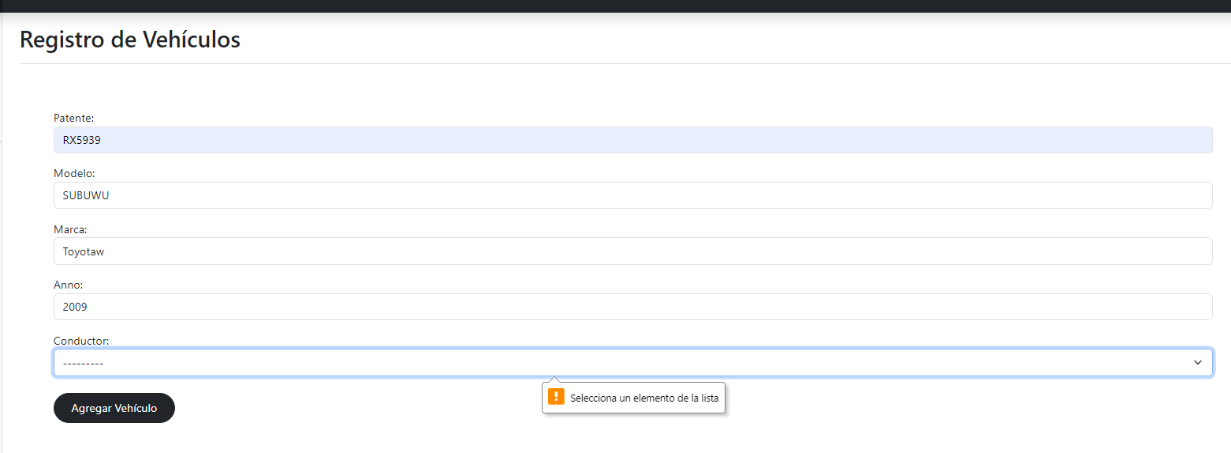


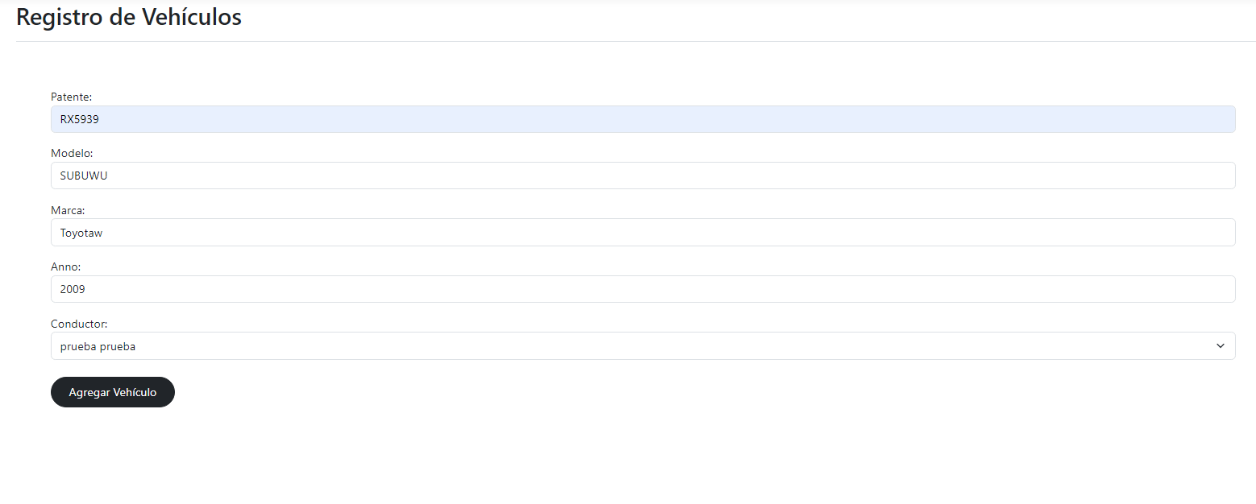
Al no poder ingresar campos en blanco, procedo a ingresar una patente sin modelo ni el resto de campos:

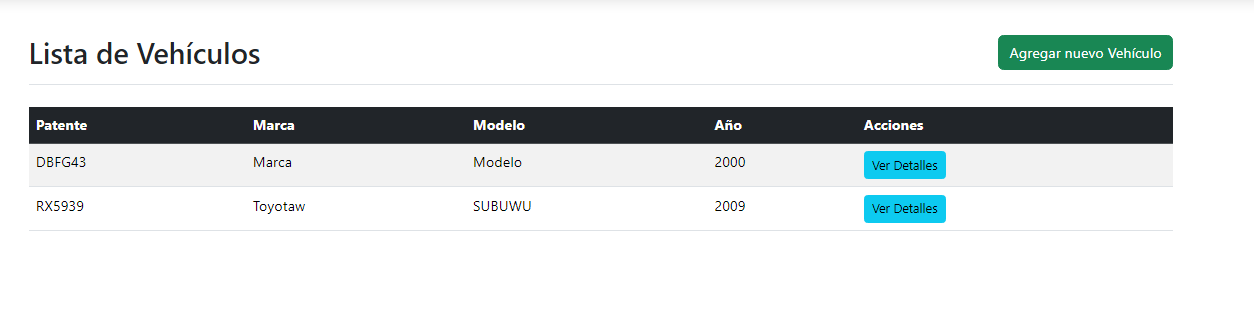
Procedo a agregar el modelo sin marca ni año

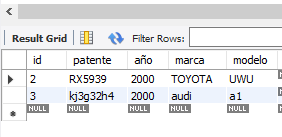
Agrego la marca:

Agregamos el año:

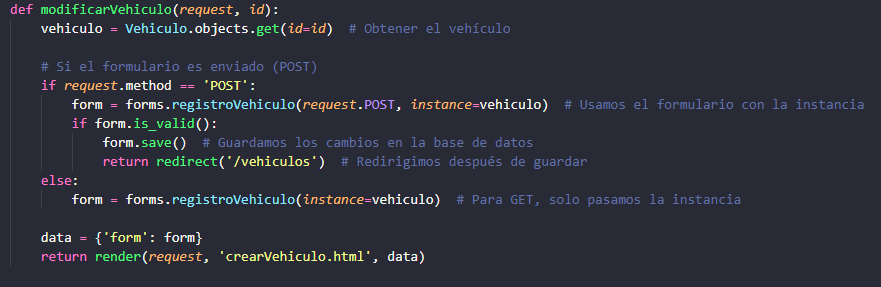
Finalmente, ya que me pide el conductor, lo agrego:

Y esto me lo deja en la lista anteriormente testeada:

**Resultado: Ingresa el vehículo a la base de datos.**



* 1. **Modificación de vehículos**
     1. Funcionalidad o código testeado:

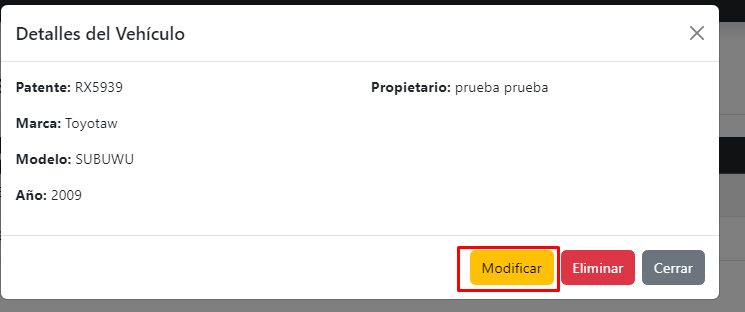


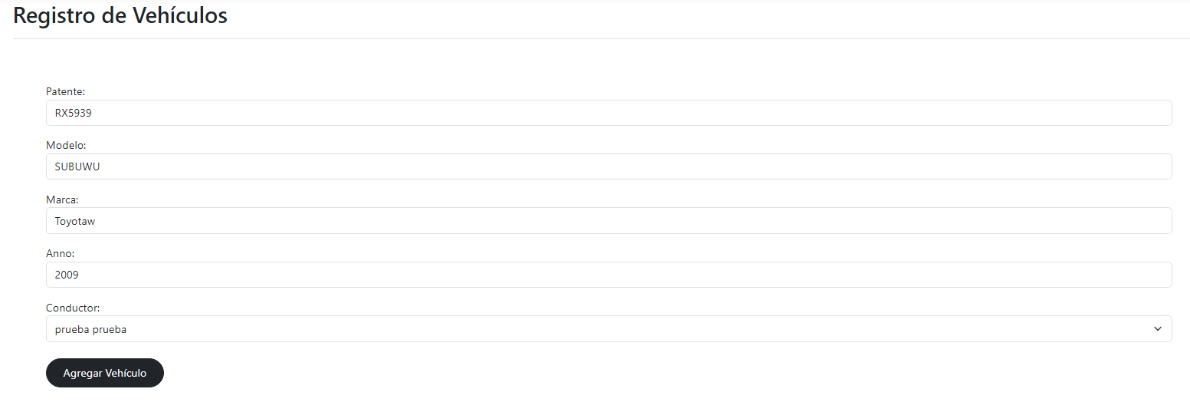
* + 1. ¿Qué busca evaluar con la prueba?:

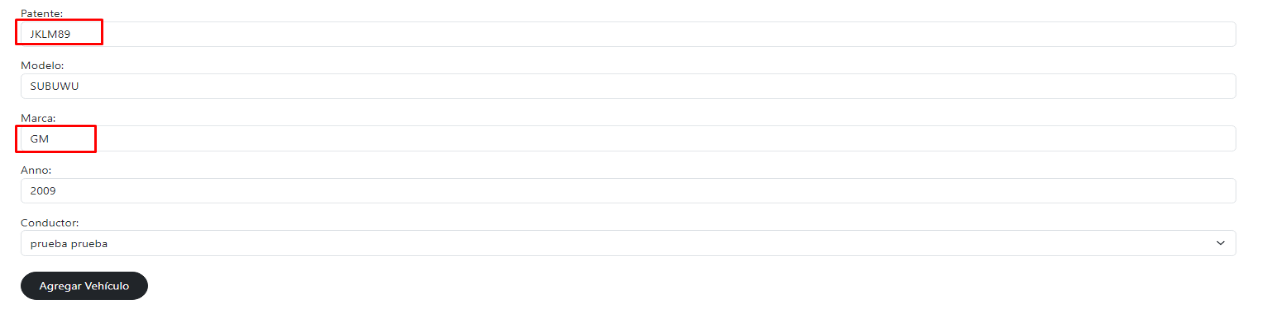
El correcto funcionamiento de que el botón “modificar” en “detalles” de cada vehículo, funcione.

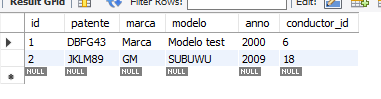
* + 1. Responsable del Testing: Esteban Astroza
    2. Resultado del Testing:

Empezamos metiéndonos al “modificar” en la lista de vehículos

Y con esto vemos que los datos, nos deja modificarlos:

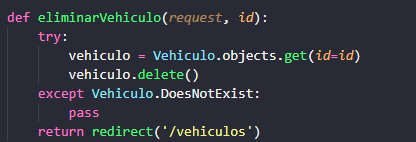
Modificamos los campos de “patente” y “marca”:

Luego verificamos si fue modificado con éxito en la base de datos:



Fue modificado correctamente.

* 1. **Eliminación de vehículos:**
     1. Funcionalidad o código testeado:

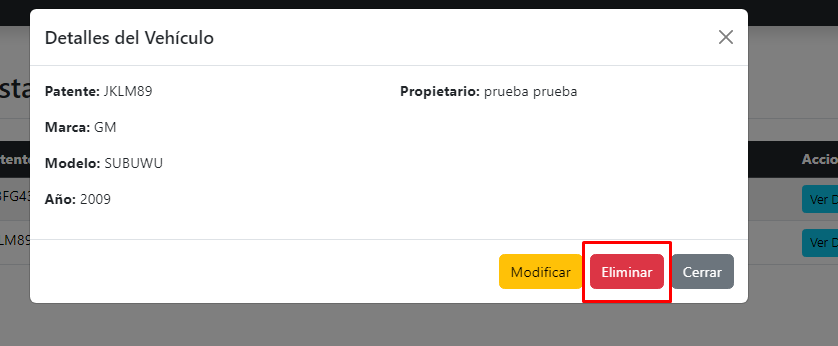


* + 1. ¿Qué busca evaluar con la prueba?:

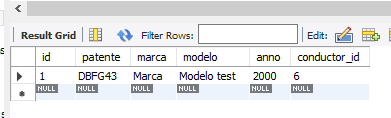
La correcta eliminación tanto como en la página web como en la base de datos de los vehículos

* + 1. Responsable del Testing: Esteban Astroza
    2. Resultado del Testing:

Empezamos metiéndonos al botón “eliminar” en la lista de vehículos

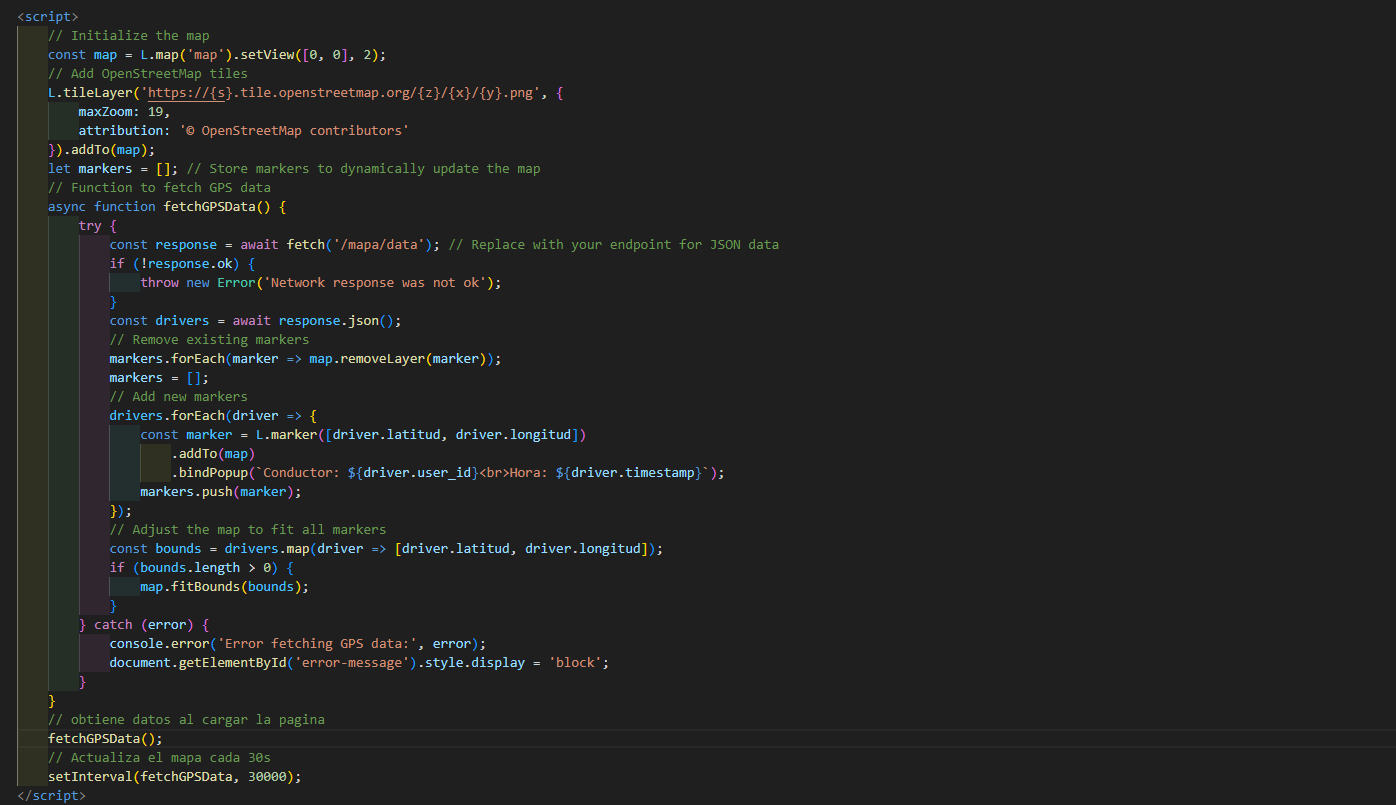


Eliminamos el vehículo anteriormente ingresado y verificamos en la base de datos



Fue eliminado correctamente.

* 1. **Mapa** 
     1. Funcionalidad o código testeado:



* + 1. ¿Qué busca evaluar con la prueba?:

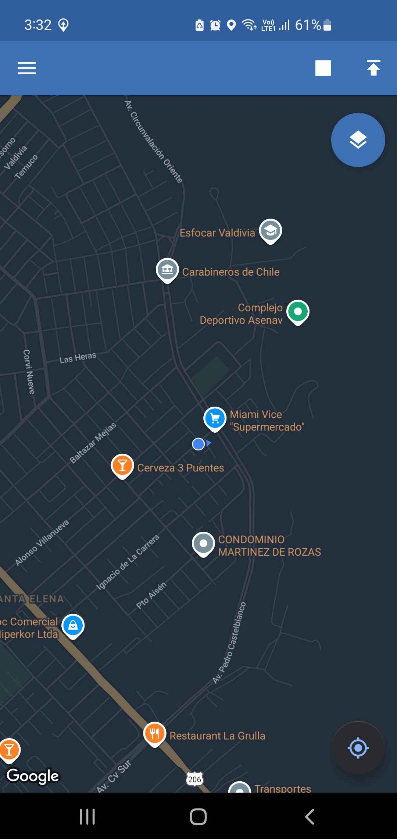
Recepción de datos GPS en el backend.

Envío de datos al frontend.

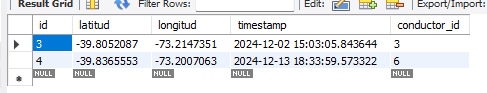
Renderización del marcador en el mapa Leaflet con las coordenadas recibidas.

* + 1. Responsable del Testing: Agustin Balcazar
    2. Resultado del Testing:

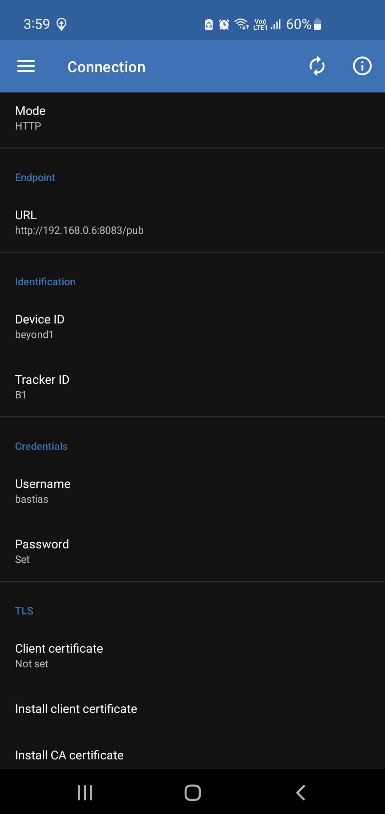
La aplicación envía los datos a la base de datos



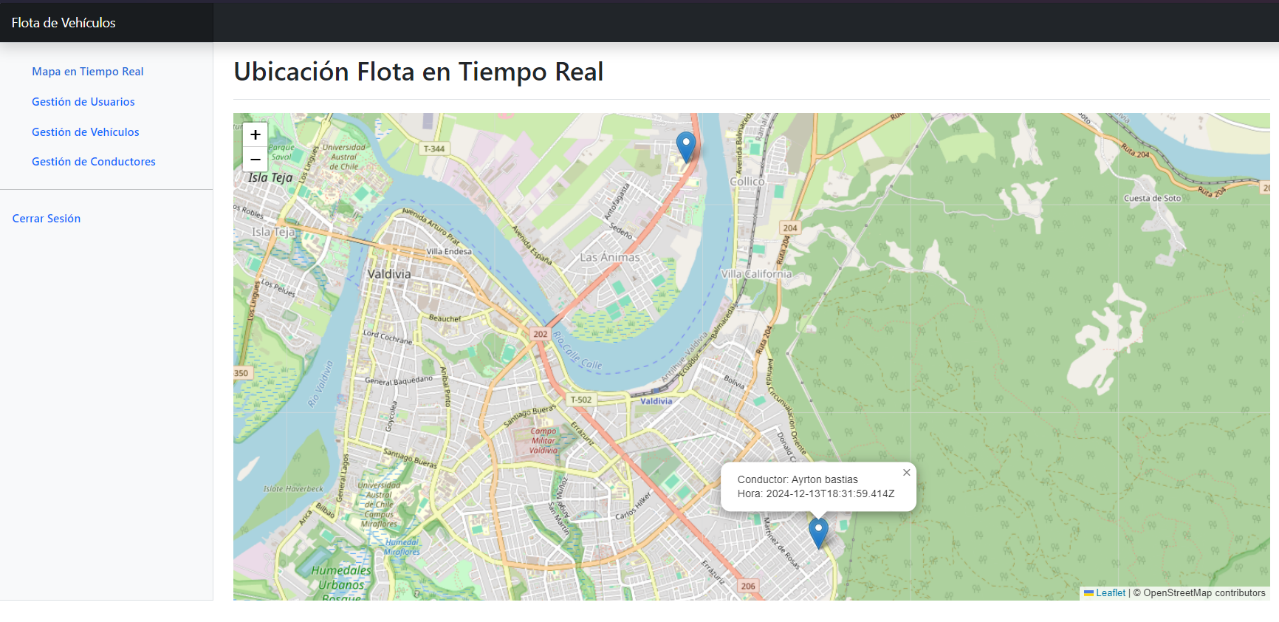
Recepción de los datos del GPS en la base de datos



Se le asigna un usuario al GPS



Se muestra la ubicación del usuario en el mapa



1. Preguntas
2. **¿Cómo pueden garantizar que el software que desarrollaron funciona de acuerdo con los requisitos y expectativas planteadas, por el cliente?**

**R:**

Garantizamos que el software desarrollado cumple con los requisitos planteados mediante la implementación y verificación de las siguientes funcionalidades clave:

* Recepción y visualización de ubicaciones en tiempo real:
  + El software es capaz de recibir direcciones y mostrarlas en un mapa en tiempo real, cumpliendo con el objetivo principal definido por el cliente.
* Gestión de vehículos y conductores:
  + Permite registrar vehículos y conductores, presentando sus datos de manera clara y organizada en el mapa para facilitar su supervisión.

Infraestructura técnica que respalda el funcionamiento global:

* La base de datos del sistema está alojada en PostgreSQL AWS, lo que garantiza que las ubicaciones y autos registrados puedan ser accedidos en tiempo real desde cualquier lugar del mundo. Aunque el software esté siendo ejecutado en diferentes ubicaciones, la infraestructura en la nube asegura que todos los usuarios conectados puedan visualizar las actualizaciones de ubicación y datos en tiempo real sin restricciones de red local.

Acciones para optimización futura:

* Aunque el sistema cumple con los requisitos actuales, estamos trabajando en la optimización del web hosting para mejorar la accesibilidad y disponibilidad general del software, garantizando un rendimiento aún más robusto y escalable.

En resumen, el software no solo cumple con los requisitos planteados, sino que, gracias a la infraestructura en la nube, permite un acceso global a la información en tiempo real, independientemente de la ubicación física de los usuarios.

1. **En su etapa de desarrollo de software, cuando realizaron pruebas, ¿Cuáles pruebas realizaron y qué detectaron? Debe indicar cómo solucionaron el problema detectado.**

Durante la etapa de desarrollo de software, se probaron todas las funcionalidades inmediatamente después de ser creadas, con el objetivo de encontrar errores y solucionarlos al instante. La mayoría de las funcionalidades no presentaron problemas, sin embargo, la funcionalidad del mapa en tiempo real sí enfrentó dificultades al intentar realizar la conexión y obtener los datos desde el dispositivo móvil. Este problema fue solucionado investigando la documentación de OwnTracks y ejecutando en el mismo entorno que la aplicación utilizando el creador de contenedores Docker.

Al consultar la documentación, se descubrió que para obtener datos en tiempo real desde la API era necesario contar con el Usuario y la ID del dispositivo del cual se desea obtener la ubicación. El Usuario fue fácil de identificar, ya que está asignado al crear el conductor en el sistema, pero obtener la ID del dispositivo fue más complejo. La aplicación móvil asigna automáticamente una ID única a cada dispositivo. Inicialmente, se consideró modificar el proceso de ingreso del conductor para requerir manualmente la ID del dispositivo. Sin embargo, esto planteaba un problema: ¿qué sucedería si el conductor utiliza más de un dispositivo?

Nuestro equipo decidió implementar una solución más automatizada. Al momento en que el conductor se conecta al sistema, su dispositivo se actualiza automáticamente y queda asignado a su usuario. Esto fue posible utilizando otra funcionalidad de la API de OwnTracks que permite visualizar los usuarios conectados y la ID de sus dispositivos.

Con estos datos correctamente gestionados, logramos acceder a la ubicación en tiempo real del conductor y actualizar su posición en el mapa de manera efectiva.

1. Bibliografía

Documentación de la API de OwnTracks:

<https://github.com/owntracks/recorder/blob/master/API.md>

Información general de OwnTracks

<https://github.com/owntracks/recorder/blob/master/README.md>

Página oficial de OwnTracks

<https://owntracks.org/booklet/>