



# Proyecto Taller Mecánico

Desarrollo del trabajo practico integrador de la catedra “Programacion avanzada 2023”. Desarrollo de sistema para un taller mecanico

Docente:

❖ Vanzetti, Juan José

Integrantes Grupo N°9:

- Gribaudo Facundo
- Sánchez Facundo
- Giménez Tomas
- Ferreyra Enzo
- Botta Joaquín
- Carnino Martin

Fecha entrega: 18/12/2023

# ÍNDICE

<b>Plan de Proyecto</b>	<b>3</b>
<b>Introducción</b>	<b>3</b>
Visión general del Proyecto	3
Entregables del Proyecto	3
Evolución del PGPS	4
Material de Referencia	4
Definiciones y Acrónimos	4
<b>Organización del Proyecto</b>	<b>5</b>
Modelo de proceso	5
Estructura organizativa	5
Límites e interfaces Organizativos	5
Responsabilidades	6
<b>Proceso de gestión</b>	<b>7</b>
Objetivos y prioridades de gestión	7
Supuestos, dependencias y restricciones	7
Gestión de riesgos	7
Mecanismos de supervisión y control	8
Plan de personal	8
<b>Proceso Técnico</b>	<b>8</b>
Métodos, herramientas y técnicas	8
Documentación de software	8
Funciones de soporte a proyectos	9
<b>Plan de Desarrollo</b>	<b>10</b>
Paquetes de trabajo	10
Dependencias	10
Requerimientos de recurso	10
Presupuesto y distribución de recursos	10
Calendario o Agenda	10
<b>Plan de pruebas de software</b>	<b>11</b>
Historial de Versiones	11
Información del proyecto	11
Aprobaciones	11
Resumen ejecutivo	11
Alcance de las pruebas	12
Elementos de pruebas	12
Funcionalidades a probar	12
Funcionalidades que no se van a probar	12
Enfoque de pruebas (estrategia)	12
Criterios de aceptación o rechazo	13
Criterios de aceptación o rechazo	13
Criterios de suspensión	13

Criterios de reanudación	13
Entregables	13
Recursos	13
Requerimientos de entorno - Hardware	13
Requerimientos de entorno - Software	14
Herramientas de pruebas requeridas	14
Personal	14
Planificación y organización	14
Procedimientos para las pruebas	14
Matriz de responsabilidad	14
Cronograma	15
Premisas	15
Dependencias y Riesgos	15
Referencias	15
<b>Documentación desarrollo - 1er Entrega</b>	<b>16</b>
Diagrama Entidad Relación (DER)	16
Primera Versión	16
Segunda Versión	17
Diagrama de Clases (DDC)	17
Primera Versión	17
Segunda Versión	19
User Story (US)	19
Gestión de Cliente	20
Gestión de registros de vehículos	21
<b>Documentación desarrollo - 2da Entrega</b>	<b>23</b>
Estimaciones US - Segunda Versión	23
Prototipos	25
Organización Tareas - Segunda Versión	30
Consideraciones MVPs	31
MVP V2	31
<b>Documentación desarrollo - 3er Entrega</b>	<b>31</b>
Requerimientos propuestos	31
Postman	32
End-Point 1	32
End-Point 2	34
End-Point 3	35
End-Point 4	36
End-Point 5	37
Codacy	39
Jconsole	40
JUnit	43
Test - Cliente	45
Test - Modelo	45



# Plan de Proyecto

## Introducción

### Visión general del Proyecto

El proyecto tiene como objetivo principal el desarrollo y la implementación de un Sistema de Gestión (integral) para el taller de autos "Autos S.A". El mismo permitirá llevar a cabo la gestión de clientes, gestión de órdenes de trabajo, gestión de stock del taller y la generación de informes y estadísticas para la toma de decisiones basadas en datos dentro de la empresa.

Todos estos aspectos serán implementados para su utilización en un software informático que se le entregará al cliente en cuestión.

El proyecto se desarrollará a través de una serie de actividades en las que se incluyen:

- Análisis de Requerimientos
- Diseño del sistema
- Implementación del sistema
- Generación de Plan de Proyecto
- Estimación de requerimientos.
- Pruebas de software
- Revisiones y control de calidad del software

Para ello se contará con una serie de recursos tanto humanos como económicos:

- Ingeniero en software (4)
- Desarrolladores (4)
- QA Tester (2)
- Diseñadores UX/UI (2)

Todo estos aspectos se encuentran abarcados dentro de un presupuesto con el que cuenta inicialmente el proyecto, de \$500.000 (ARS)

### Entregables del Proyecto

Versión	Descripción	Entregable	Fecha entrega	Lugar entrega
1.0	Se entrega la funcionalidad inicial que permite la gestión de vehículos, marca, modelo, clientes y técnico.	Gestión de vehículos, marca, modelo, clientes y técnicos.	11/09/2023	A definir.
1.1	Se implementan las user stories solicitadas para la segunda versión del MVP	Gestión de Orden de Trabajo y Servicios	09/10/2023	A definir.

## Evolución del PGPS

Este apartado no aplica.

## Material de Referencia

<u>Título</u>	<u>Informe N°</u>	<u>Autor</u>	<u>Organizacion Publicada</u>	<u>Referencia</u>
Norma IEEE 1058.1	1	Cobos Lomeli Manuel Alejandro  Lopez Rivera Jose Miguel  Hernande Hernandez Aaron  Prof. Margarita Maria de Lourdes Sanchez	Universidad Autónoma Metropolitana	<a href="http://aniei.org.mx/paginas/uam/CursoAI/IEEE_rep.pdf">http://aniei.org.mx/paginas/uam/CursoAI/IEEE_rep.pdf</a>

## Definiciones y Acrónimos

<u>Acrónimo</u>	<u>Definición</u>
PGPS	Plan para la Gestión de Proyectos Software
DER	Diagrama de Entidad - Relación
QA	Quality Assurance
Scrum	Scrum es un marco de trabajo ágil que se enfoca en la entrega iterativa y en la gestión eficaz de proyectos
Scrum Master	Es un rol donde su principal responsabilidad es facilitar y liderar la implementación de Scrum en un equipo o proyecto.
Product Backlog	Es listado de tareas ha realizar en el

	proyecto
UX	Experiencia de usuario
UI	Interfaz de usuario

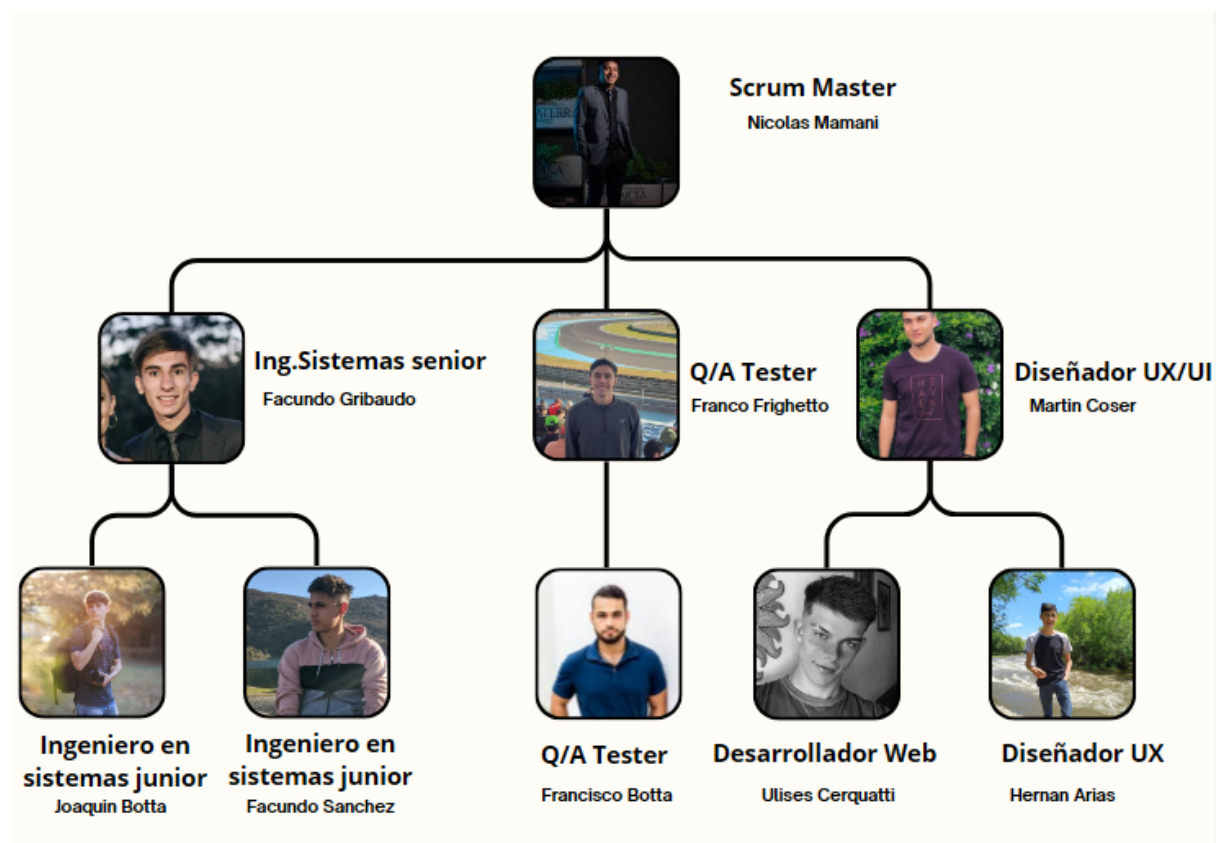
## Organización del Proyecto

### Modelo de proceso

Este apartado no aplica.

### Estructura organizativa

A continuación se presenta una diagramación de la estructura organizacional interna encargada de la gestión del proyecto en cuestión.



### Límites e interfaces Organizativos

Las entidades participantes del proyecto son:

- Organización encargada del proyecto
- Organización cliente

La **organización encargada del proyecto** se limitará a las actividades relacionadas a la comunicación constante con el cliente, la captación de requerimientos y el desarrollo e implementación de los mismos.

Por otro lado, llevará a cabo la autoevaluación utilizando como base las retrospectivas dadas por el cliente y de esta forma desempeñará una serie de actividades para la reorientación del proyecto, en caso de que fuese necesario.

Por otra parte el **cliente** limita su participación en el proyecto a las actividades de informe de requerimientos que desea, así como también a la mantención y/o actualización de los mismos en caso de que fuese necesario. A su vez y siempre que lo considere pertinente podrá estar presente en las reuniones Daly llevadas a cabo diariamente por la organización del proyecto para conocer distintos aspectos, como por ejemplo el avance del mismo.

Finalmente será el encargado de evaluar los posibles incrementos de producto que la parte desarrolladora del software le presente y realizar una retroalimentación de la misma para poder realizar las mejoras que considere necesarias, en caso de que lo fuese.

## Responsabilidades

Actividad	Responsable	Consultado	Informado
Comunicación con el Cliente	Equipo de Analistas	Equipo de Desarrolladores, QA	Equipo de Ingenieros
Captación de Requerimientos	Equipo de Analistas	Equipo de Ingenieros	Equipo de Ingenieros
Desarrollo e implementación	Equipo de Desarrolladores	Equipo de QA	Equipo de Analistas, Ingenieros
Autoevaluación y Retrospectivas	Equipo de Desarrolladores	Equipo de QA, Analistas	Equipo de Ingenieros
Mantenimiento y actualización	Cliente	Equipo de Analistas	Equipo de Desarrolladores, QA, Ingenieros
Evaluación de incrementos	Cliente	Equipo de Desarrolladores	Equipo de QA, Analistas, Ingenieros



## Proceso de gestión

### Objetivos y prioridades de gestión

Durante el desarrollo del proyecto se tendrán en cuenta los siguientes objetivos y prioridades:

- **Entregar un sistema de gestión funcional**: Lo primordial es desarrollar un sistema que cumpla con todos los requisitos especificados.
- **Satisfacción del cliente**: se debe asegurar que el software cumpla con las expectativas y necesidades del cliente.
- **Calidad del software**: Es muy importante para la gerencia la fiabilidad, el rendimiento, la seguridad y la capacidad de mantenimiento del software.
- **Cumplimiento de plazos**: Mantenerse dentro del cronograma planificado es crucial para la entrega puntual y para evitar retrasos costosos.
- **Colaboración del equipo**: Fomentar la colaboración y la comunicación efectiva dentro del equipo de desarrollo para mantener la eficiencia y la moral del equipo.

A medida que transcurra el proyecto y nuestros objetivos y prioridades cambien/evolucionen los iremos identificando en el presente apartado.

### Supuestos, dependencias y restricciones

Supuestos:

- El cliente está predispuesto para la comunicación continua.
- Disponibilidad de recursos para iniciar el proyecto.
- Aceptación de cambios.

Dependencias:

- Adquisición de licencias.
- Capacitación de personal.

Restricciones:

- Tiempo y recursos.
- Normativas del país.

### Gestión de riesgos

Este apartado no aplica

## Mecanismos de supervisión y control

Este apartado no aplica

## Plan de personal

Este apartado no aplica

## Proceso Técnico

### Métodos, herramientas y técnicas

Para el análisis y desarrollo de este proyecto hemos decidido aplicar métodos basados en ideologías ágiles, lo cual nos permitirá llevar a cabo el mismo, haciendo utilización del marco de trabajo definido por “SCRUM”. Por otra parte haremos utilización de la notación UML para la representación y comprensión del dominio, específicamente los “Diagramas de Clase” que provee dicha notación, así como cualquier otro que nos sea de ayuda durante el transcurso del proyecto.

Para la implementación haremos utilización del lenguaje de programación “Java” version 20.x, acompañado de distintas tecnologías como:

- **Framework Angular, HTML y/o CSS** (front end)
- **Framework Spring boot** (back end)
- **Postgresql y/o MySQL** (base de datos)

Por otra parte, utilizaremos distintas técnicas que nos permita identificar los requerimientos del cliente y que podamos reconocer la complejidad de los mismos para así lograr una correcta organización en las actividades a desarrollar. Las técnicas en cuestión son:

- **Historias de usuario:** Técnica utilizada para la captura y representación de los requerimientos.
- **Poker Planning:** Técnica para llevar a cabo la estimación de los requerimientos y conocer la complejidad de los mismos.

## Documentación de software

- **Product Backlog:** es un artefacto para la visualización de las necesidades individuales del product owner y donde el equipo Scrum puede saber el alcance del proyecto.

### Gestión de cliente

- Registrar cliente
- Consultar cliente
- Programar cita con cliente

- Visualizar calendario
- Enviar confirmaciones de citas

#### Gestión de órdenes de trabajo

- Crear orden de trabajo
- Asignar técnico
- Consultar órdenes de trabajo
- Consultar estado de orden de trabajo

#### Gestión de Stock

- Registrar entrada de repuesto
- Registrar salidas del repuesto

#### Informes y estadísticas

- Generar informe de ventas periódicas
- Generar servicios más solicitados

#### Gestión de vehículos

- Registrar vehículo
- Consultar vehículos
- Registrar marca

- **Diagrama de entidad - relación** : es un tipo de diagrama de flujo que ilustrar entidades, personas, objetos o conceptos que intervienen en el sistema y se relacionan entre sí. Su uso principal será para el diseño y depuración de base de datos.

- **Diagrama de paquetes**

Cada paquete representa una unidad de gestión del product backlog, con las historias de usuario que se tienen en cuenta



## Funciones de soporte a proyectos

Este apartado no aplica

## Plan de Desarrollo

### Paquetes de trabajo

Este apartado no aplica.

### Dependencias

Este apartado no aplica.

### Requerimientos de recurso

Este apartado no aplica.

### Presupuesto y distribución de recursos

Este apartado no aplica

### Calendario o Agenda

Este apartado no aplica

# Plan de pruebas de software

## Historial de Versiones

Fecha	Versión	Autor	Organización	Descripción
24/11/2023	1.0	Joaquin Botta		
18/12/2023	1.1	Joaquin Botta		

## Información del proyecto

Empresa / Organización	Grupo Desarrollo
Proyecto	Taller Mecánico
Fecha de preparación	18/12/2023
Cliente	Autos S.A
Patrocinador principal	Autos S.A
Gerente / Líder de proyecto	Nicolás Mamani
Gerente / Líder de pruebas de software	Nicolás Mamani

## Aprobaciones

Nombre y Apellido	Cargo	Departamento u organización	Fecha	Firma
Franco Frighetto	QA tester	QA Tester	24/11/2023	FG

## Resumen ejecutivo

El plan de pruebas tiene como objetivo garantizar la calidad y fiabilidad del software diseñado para el sistema de gestión del taller mecánico, verificando que todas las funciones se ejecuten según lo esperado y se cumplan los requerimientos establecidos.

## Alcance de las pruebas

### Elementos de pruebas

Alto Nivel (Áreas Funcionales):

1. Gestión de Clientes:
  - Registro de clientes
  - Almacenamiento correcto en la base de datos
2. Gestión de modelos:
  - Registro de modelos
  - Validación de campos obligatorios
  - Almacenamiento correcto en la base de datos

Cada uno de estos elementos, ya sea a nivel de área funcional o detallado, requeriría pruebas específicas para asegurar su correcto funcionamiento, evitando errores y garantizando la experiencia óptima para los usuarios del sistema de gestión del taller mecánico.

### Funcionalidades a probar

Gestión de clientes: Se verificará la correcta creación y almacenamiento de objetos clientes dentro de la entidad "Cliente" de la base de datos.

Gestión de modelos: Se verificará la correcta creación y almacenamiento de los objetos modelo dentro de la entidad "Modelo" de la base de datos.

Por otro lado se verificará que cada uno de estos objetos se encuentren vinculados a un objeto de la entidad "Marca" para satisfacer las reglas de negocio planteadas en un principio del proyecto.

### Funcionalidades que no se van a probar

No aplica.

### Enfoque de pruebas (estrategia)

Tipos de pruebas:

- **Pruebas Funcionales**: Verificación de las funciones clave del sistema, incluyendo el registro y actualización de clientes, gestión del historial de servicios, generación de informes, entre otros.
- **Pruebas de Rendimiento**: Evaluación del rendimiento del sistema bajo diferentes cargas de trabajo para asegurar que responda eficientemente.

## Criterios de aceptación o rechazo

### Criterios de aceptación o rechazo

Completar 100% de pruebas unitarias, cierto porcentaje de casos exitosos, cobertura de todos los componentes y líneas de código, porcentaje de defectos corregidos, entre otros.

### Criterios de suspensión

Los criterios de suspensión de pruebas se aplican en situaciones donde la continuación de las pruebas puede ser contraproducente o inviable. Estos criterios se activan en presencia de defectos críticos que impacten la funcionalidad central del sistema, si un porcentaje significativo de casos de prueba (por ejemplo, más del 50%) fallan por una causa común, o cuando errores impiden la ejecución de pruebas adicionales. Además, se considera la suspensión si ocurren cambios drásticos en los requisitos o la arquitectura del sistema durante el proceso de pruebas, lo que invalidará la efectividad de las pruebas existentes. Estos criterios permiten tomar decisiones prudentes para asegurar la eficiencia y validez de las pruebas realizadas

### Criterios de reanudación

Los criterios para reanudar las pruebas se establecen una vez que se hayan corregido los defectos críticos que causaron la suspensión. Se reanudarán si el porcentaje de casos fallidos se reduce a un nivel aceptable, las funcionalidades clave están nuevamente disponibles y si los cambios en requisitos o arquitectura han sido validados sin riesgos significativos. Estos criterios garantizan que las pruebas se reanuden de manera efectiva y bajo condiciones que aseguren su validez y eficiencia.

## Entregables

- Documento plan de entrega
- Descripción de test definidos, realizados y resultados de los mismos.

## Recursos

### Requerimientos de entorno - Hardware

- Bases de Datos: Un entorno de base de datos compatible y adecuado para almacenar y gestionar la información del sistema, con respaldo para restauración en caso de pruebas críticas.
- Equipos de Pruebas para Testers: Computadoras personales con especificaciones mínimas que permitan la ejecución de pruebas, simulando condiciones reales de uso del software.

- Conectividad de Red: Acceso estable a la red local o remota

## Requerimientos de entorno - Software

- Sistema Operativo: Versiones compatibles del sistema operativo necesarias para ejecutar el software de pruebas y el sistema de gestión del taller mecánico.
- Software del Sistema de Gestión: Acceso al software del sistema de gestión del taller mecánico en un entorno de pruebas para ejecutar pruebas funcionales, de rendimiento y seguridad.
- Acceso a Bases de Datos de Pruebas: Acceso a bases de datos específicas para pruebas que reflejan el entorno de producción pero que permitan manipulación sin afectar los datos reales.

## Herramientas de pruebas requeridas

Herramientas de pruebas: Instalación de herramientas de prueba adecuadas para llevar a cabo pruebas funcionales, pruebas de carga, herramientas de automatización, si es aplicable, para agilizar el proceso de pruebas.

## Personal

1. Scrum Master
2. QA Tester

## Planificación y organización

### Procedimientos para las pruebas

- Pruebas Funcionales: Verificación de que cada función del sistema cumpla con los requisitos establecidos. Se pueden emplear técnicas como casos de uso, escenarios y matrices de trazabilidad para garantizar que todas las funciones se prueban exhaustivamente.
- Pruebas de Rendimiento: Evaluación del desempeño del sistema bajo diferentes cargas de trabajo para asegurar que responda de manera eficiente y no tenga problemas de capacidad.

## Matriz de responsabilidad

Tareas	Responsable	Aprobador
Elaboración y ajuste del Plan de Pruebas de Aceptación	Francisco Botta	Franco Frighetto
Elaboración y ajuste de los scripts de pruebas		
Elaboración y ajuste de los casos de pruebas		



Revisión y aprobación del Plan de Pruebas de Aceptación Revisión y aprobación de los scripts de pruebas Revisión y aprobación de los casos de pruebas	Francisco Botta	Franco Frighetto
Elaboración de los datos de prueba	Francisco Botta	Franco Frighetto
Suministro del ambiente de pruebas	Francisco Botta	Franco Frighetto
Instalación del ambiente de pruebas	Francisco Botta	Franco Frighetto
Ejecución de las pruebas de validación	Francisco Botta	Franco Frighetto
Ejecución de las pruebas de aceptación	Francisco Botta	Franco Frighetto
Evaluación de las pruebas	Francisco Botta	Franco Frighetto
Reporte de avance de las pruebas	Francisco Botta	Franco Frighetto
Reporte sumario de pruebas	Francisco Botta	Franco Frighetto

## Cronograma

1. **Elaboración y ajuste del Plan de Pruebas de Aceptación** (Semana 1-2)
2. **Elaboración y ajuste de los scripts de pruebas** (Semana 3-4)
3. **Elaboración y ajuste de los casos de pruebas** (Semana 5-6)
4. **Revisión y aprobación del Plan de Pruebas de Aceptación** (Semana 7)
5. **Revisión y aprobación de los scripts de pruebas** (Semana 8)
6. **Revisión y aprobación de los casos de pruebas** (Semana 9)
7. **Elaboración de los datos de prueba** (Semana 10)
8. **Suministro del ambiente de pruebas** (Semana 11)
9. **Instalación del ambiente de pruebas** (Semana 12)
10. **Ejecución de las pruebas de validación** (Semanas 13-15)
11. **Ejecución de las pruebas de aceptación** (Semanas 16-18)
12. **Evaluación de las pruebas** (Semanas 19-20)
13. **Reporte de avance de las pruebas** (Semana 21)
14. **Reporte sumario de pruebas** (Semana 22)

## Premisas

No aplica.

## Dependencias y Riesgos

No aplica.

## Referencias

- Plan de proyecto.
- Especificaciones de requerimientos.
- Diseño general.
- Diseño detallado.
- Procedimientos y estándares de desarrollo.
- Procedimientos y estándares de pruebas.

# Documentación desarrollo - 1er Entrega

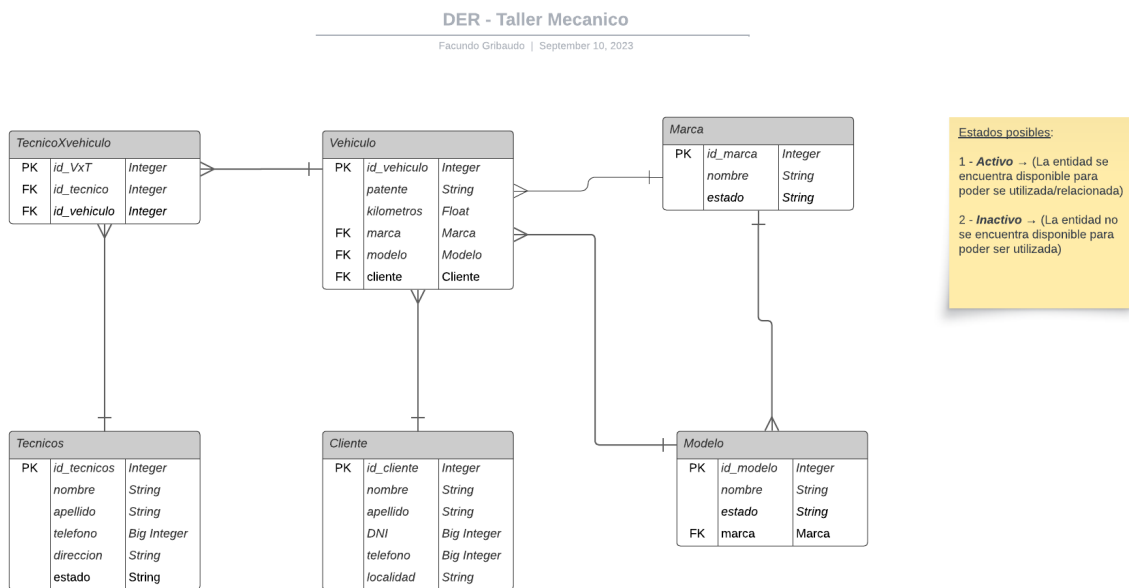
A continuación se presenta toda la documentación desarrollada para la primera versión del sistema del Taller Mecánico. La misma será actualizada a medida que el proyecto evolucione.

## Diagrama Entidad Relación (DER)

### Primera Versión

El diagrama presentado a continuación representa la estructura lógica de la base de datos MySQL sobre la cual funciona el sistema informático.

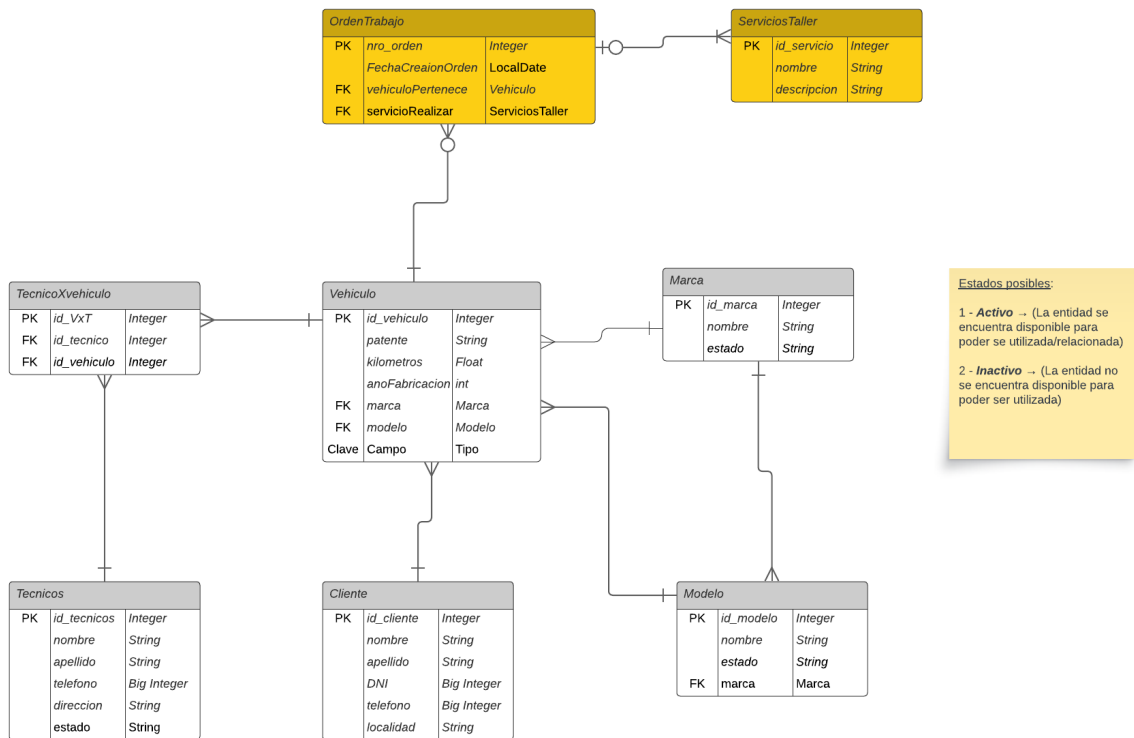
En él podremos ver las distintas entidades existentes así como también sus atributos, tipos de datos, claves primarias y foráneas de las distintas entidades y las relaciones con otras entidades.



**Adjuntamos link para una mejor visualización:**

[https://drive.google.com/file/d/1zikUL-QJFPTZ1DikrFdCoXha\\_OD0payK/view?usp=drive\\_link](https://drive.google.com/file/d/1zikUL-QJFPTZ1DikrFdCoXha_OD0payK/view?usp=drive_link)

## Segunda Versión



En la imagen adjunta se resaltan las tablas que fueron implementadas para la versión en cuestión (V2)

**Adjuntamos link para una mejor visualización:**

[https://drive.google.com/file/d/1d\\_TjjgD4sqxrpmzbow1lfF0yBbiRPilmD/view?usp=sharing](https://drive.google.com/file/d/1d_TjjgD4sqxrpmzbow1lfF0yBbiRPilmD/view?usp=sharing)

## Diagrama de Clases (DDC)

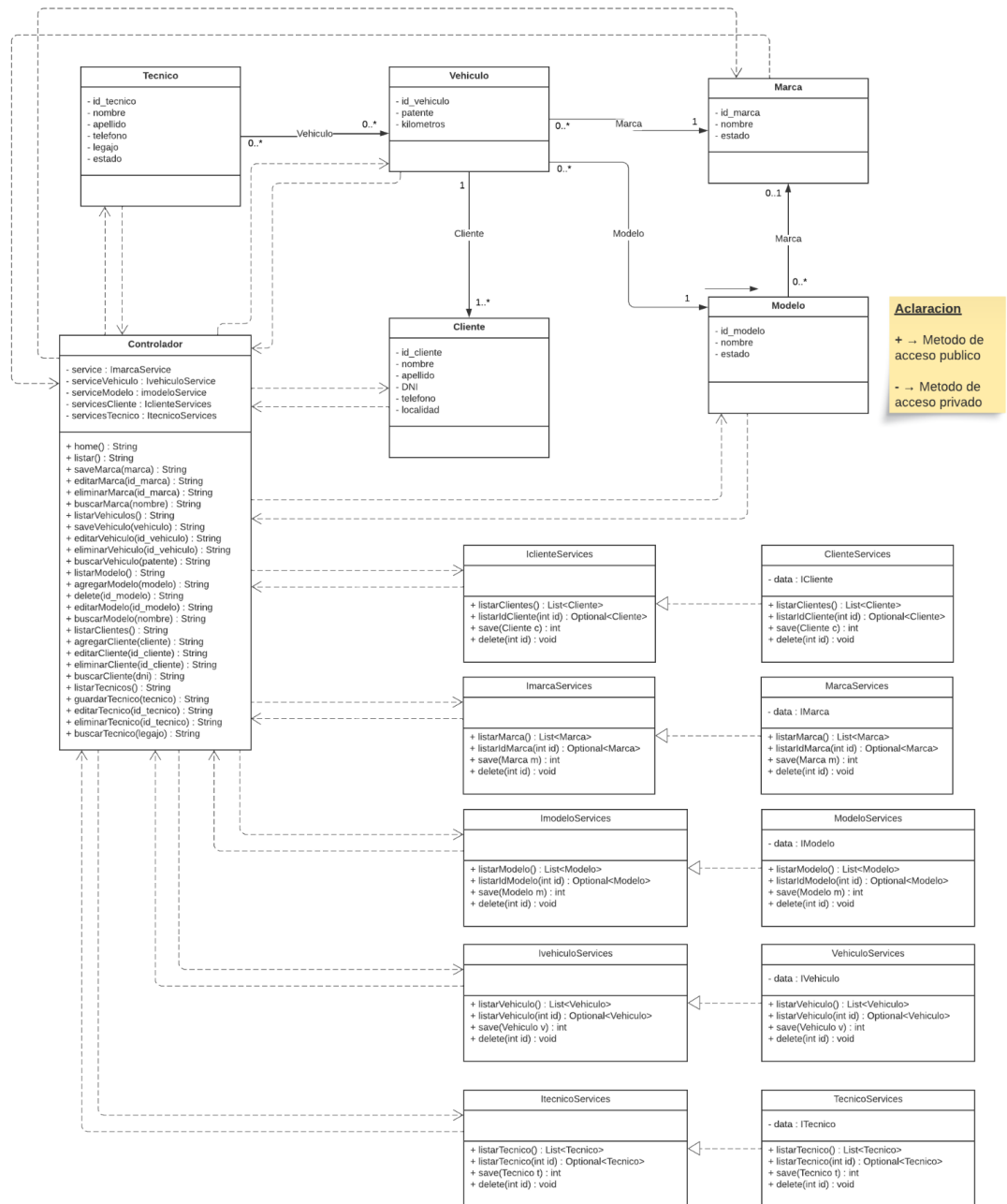
### Primera Versión

El diagrama a continuación representa el dominio del taller implementado en esta primera versión del software informático.

Podemos observar las distintas clases existentes en el dominio correspondiente, así como sus atributos, funcionalidades (métodos), enlaces, navegabilidad y multiplicidad correspondiente.

DDC - Taller mecanico PA 2023

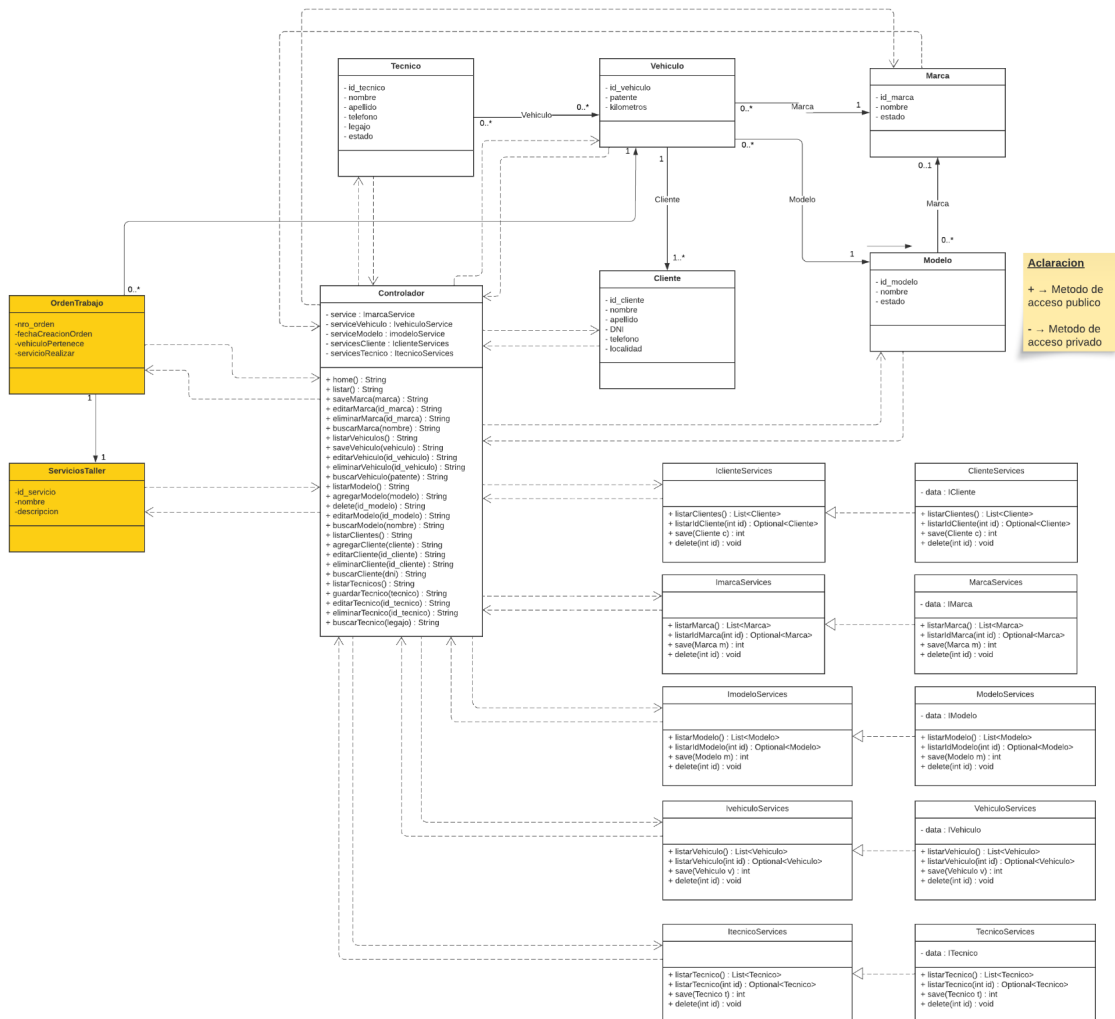
Facundo Gribaudo | September 11, 2023



**Adjuntamos link para una mejor visualización:**

[https://drive.google.com/file/d/1kxQq0jI0wWhixEkxzkz9uzjxB7TaX7E5/view?usp=drive\\_link](https://drive.google.com/file/d/1kxQq0jI0wWhixEkxzkz9uzjxB7TaX7E5/view?usp=drive_link)

## Segunda Versión

DDC - Taller mecanico PA 2023  
Facundo Gribaudo | October 9, 2023

Adjuntamos link para una mejor visualización:

<https://drive.google.com/file/d/18n9qF3Gagnw6ldKfqctjUMLStn1hDiEo/view?usp=sharing>

## User Story (US)

A continuación se presentan las User Story identificadas durante el análisis y se especifican aquellas que fueron implementadas en la primera versión del software.

**Roles identificados:**

- Usuario
- Tecnico
- Gerente
- Recepcionista
- Administrador

### **US Identificadas:**

- Usuario administrativo
  - Registrar cliente
  - Agregar informacion cliente
  - Filtrar lista de clientes
  - Programar citas para los clientes
  - Visualizar calendario
  - Visualizar historial de servicios de cliente
  - Enviar confirmaciones de citas
  - Recibir notificaciones de clientes
  - Crear orden de trabajo
  - Asignar técnicos a órdenes de trabajo
  - Generar facturas
  - Gestionar stock de taller
  - Visualizar historial de stock taller
  - Recibir alertas de stock taller
  - Generar reportes de stock taller
  - Buscar repuestos taller.
  - Recibir sugerencia de repuestos
  - Generar informes de ventas
  - Generar estadísticas de rendimiento personal
- Tecnico
  - Acceder a las órdenes de trabajo asignadas
  - Buscar órdenes de trabajo.
- Gerente
  - Realizar seguimiento de las órdenes
- Recepcionista
  - Registrar detalles de vehiculos
- Administrador
  - Administrar marcas y modelos de vehículos
  - Administrar marcas de automóviles.

En esta versión del software implementamos las siguientes user story:

### **Gestion de Cliente**

#### **MVP Versión N°1**

#### **Historia de usuario**

Como usuario, quiero registrar los datos de un nuevo cliente en el sistema, incluyendo nombre, dirección e información de contacto, para mantener un historial completo.

#### **Criterios de aceptación**

- El sistema debe permitir el registro de nuevos clientes, solicitando información como nombre, dirección, número de teléfono y dirección de correo electrónico

- Debe ser posible acceder y modificar la información del cliente en cualquier momento
- Los datos del cliente deben estar almacenados de manera segura y protegidos contra accesos no autorizados

### **Historia de usuario**

Como usuario, quiero poder buscar y filtrar la lista de clientes según los criterios, como nombre del cliente, para acceder rápidamente a la información necesaria.

#### **Criterios de aceptación**

- El sistema debe ignorar la diferencia entre mayúsculas y minúsculas a la hora de realizar las búsquedas.

### **MVP Versión N°2**

#### **Historia de usuario**

Como usuario, quiero poder registrar los datos de un nuevo cliente en el sistema, incluyendo nombre, dirección, información de contacto y registros de servicios anteriores, para mantener un historial completo.

#### **Criterios aceptación**

- El sistema debe permitir el registro de nuevos clientes, solicitando información como nombre, dirección, número de teléfono y dirección de correo electrónico
- Debe ser posible acceder y modificar la información del cliente en cualquier momento

#### **Historia de usuario**

Como usuario, quiero tener la capacidad de agregar información adicional del cliente, como su licencia de conducir o el modelo de su automóvil, para tener detalles relevantes a mano.  
Historia de usuario

#### **Historia de usuario**

Como usuario, quiero poder buscar y filtrar la lista de clientes según los criterios, como nombre o fecha de última visita, para acceder rápidamente a la información necesaria.

## Gestión de registros de vehículos

### **MVP Versión N°1**

#### **Historia de usuario**

Como recepcionista, quiero poder registrar los detalles de los vehículos de los clientes en el sistema, incluyendo la marca, modelo, patente y kilometraje, para mantener un registro completo de cada vehículo.

#### **Criterios de aceptación**

- Al ingresar un nuevo vehículo al taller, el sistema debe permitir registrar la marca, modelo, patente y cualquier otra información relevante.
- Debe ser posible asociar el vehículo con el cliente correspondiente en el sistema.
- El sistema debe mostrar una lista de todos los vehículos registrados, permitiendo buscar y filtrar por patente.

### **Historia de usuario**

Como administrador del sistema quiero poder cargar y mantener una lista de marcas de automóviles en el sistema, para poder asociar los modelos correspondientes y tener una clasificación adecuada de los vehículos.

#### **Criterios de aceptación**

- El sistema debe proporcionar una interfaz para cargar y administrar una lista de marcas de automóviles, solicitando el nombre de la marca y cualquier información adicional relevante.
- Debe ser posible agregar, modificar y eliminar marcas de automóviles en el sistema según sea necesario.
- El sistema debe garantizar que no haya duplicados en la lista de marcas de automóviles.

### **Historia de usuario**

Como administrador del sistema, quiero poder asociar modelos de automóviles a las marcas correspondientes en el sistema, para tener una clasificación más detallada de los vehículos.

#### **Criterios de aceptación**

- El sistema debe permitir asociar modelos de automóviles a las marcas existentes en el sistema, solicitando el nombre del modelo y la marca a la que pertenece.
- Debe ser posible agregar, modificar y eliminar modelos de automóviles en el sistema según sea necesario.
- El sistema debe garantizar que no haya duplicados en la lista de modelos de automóviles.

## **MVP Versión N°2**

### **Historia de usuario**

Como usuario, quiero poder crear una orden de trabajo para cada vehículo que ingresa al taller, registrando los datos del cliente, los servicios solicitados y cualquier otra información relevante

#### **Criterios de aceptación**

- El sistema debe permitir la creación de órdenes de trabajo, solicitando la información del vehículo, como la marca, modelo, año y patente.
- Debe ser posible registrar los servicios solicitados por el cliente, así como cualquier información adicional relevante, como problemas específicos o instrucciones especiales
- El sistema debe generar automáticamente un número de orden de trabajo único para cada registro.

Las User Story mencionadas anteriormente, son las que se encuentran implementadas en la versión 1 y 2 (presente), del correspondiente software informático. El mismo cumple con lo descrito en cada una de las user story junto a sus criterios de aceptación.

En las próximas versiones del software, se irán implementando las demás que sean solicitadas por el cliente/dominio.



# Documentación desarrollo - 2da Entrega

## Estimaciones US - Segunda Versión

Como primera forma de organización que llevamos a cabo para el desarrollo de la segunda etapa, fue llevar a cabo una estimación de las User Stories propuestas para conocer la complejidad de las mismas y las tareas que necesitamos para cumplir con las mismas.

La estimación en cuestión es la siguiente:

ID US	Enunciado Historia	Esfuerzo	Prioridad	Comentarios
1	Como usuario, quiero poder registrar los datos de un nuevo cliente en el sistema, incluyendo nombre, dirección, información de contacto y registros de servicios anteriores, para mantener un historial completo.	3	Alta	Decidimos asignarle un valor de "3" debido a que consiste en implementar un ABM de una entidad que hace utilización de datos sencillos sin presentar ningún grado de dificultad significativo y que poseemos todos los conocimientos técnicos para implementar la funcionalidad solicitada. Conlleva trabajo enfocado en realizar las validaciones necesarias en cada uno de los campos.
2	Como usuario, quiero tener la capacidad de agregar información adicional del cliente, como su licencia de conducir o el modelo de su automóvil, para tener detalles relevantes a mano. Historia de usuario	1	Baja	Decidimos asignarle un valor de "1" debido a que no presenta complejidad alguna ya que simplemente requiere agregar campos a entidades ya existentes para poder cargar los nuevos datos solicitados y realizar las verificaciones correspondientes de los mismos.
3	Como usuario, quiero poder buscar y filtrar la lista de clientes según los criterios, como nombre o fecha de última visita, para acceder rápidamente a la información necesaria.	1	Baja	Decidimos asignarle un valor de "1" debido a que no presenta complejidad alguna ya que únicamente debemos filtrar de forma correcta los datos cargados en la base de datos para poder mostrar los clientes con la información solicitada.

4	Como usuario, quiero poder crear una orden de trabajo para cada vehículo que ingresa al taller, registrando los datos del cliente, los servicios solicitados y cualquier otra información relevante	3	Alta	Decidimos asignarle un valor de "3" debido a que representa la implementación de un ABM de una entidad la cual hace utilización de datos sencillos, los cuales ya se encuentran almacenados en la base de datos y únicamente necesitamos realizar las consultas correspondientes a la misma para recuperarlos y trabajar con ellos. Representa un esfuerzo a la hora de definir correctamente las relaciones ya realizar las consultas necesarias para recuperar los datos que realmente necesitamos, de forma correcta.
---	---	---	------	--

## Prototipos

### Funcionalidad - Servicio

A continuación presentamos los distintos prototipos diseñados para las nuevas secciones consideradas en la segunda versión del MVP.

#### Descripción de las pantallas:

En la pantalla “Inicio” del software encontraremos las distintas opciones que nos brinda el sistema. En esta segunda etapa hemos implementado específicamente la funcionalidades que permiten gestionar “Servicios” y “Órdenes de trabajo” que maneja el taller.



Ingresando a la opción “Servicios” nos encontraremos con la funcionalidad requerida, la cual permite registrar un nuevo servicio que provea el taller, simplemente completando los campos solicitados (“Nombre” y “Descripción servicio”). Una vez completado lo registramos haciendo utilización del botón correspondiente (“Registrar Servicios”)



Una vez registrado el nuevo servicio se actualizará automáticamente una tabla que muestra todos los servicios con los que cuenta el taller.



Cada uno de los servicios cargados editados y o eliminados en caso de que sea necesario



En caso de que seleccionemos la opción de eliminar un servicio, el sistema nos pedirá confirmación para realizar dicha acción. En caso de confirmar la acción, el correspondiente registro será eliminado

localhost:8080 dice

Estas seguro de eliminar?

Acceptar Cancelar

En caso de utilizar la opción de editar, nos redirigirá a un nuevo apartado del sistema que nos permitirá modificar el campo que necesitamos.

Una vez modificado el dato que necesitamos, seleccionamos la opción de “Actualizar” y veremos los cambios reflejados en la tabla de registros correspondiente (mostrada anteriormente)

En caso de seleccionar la opción “Cancelar” no llevaríamos a cabo ninguna modificación.



Formulario de edición de servicio. El título es "EDITAR SERVICIO". Hay dos campos de texto: "CAMBIO ACEITE" y "DESCRIPCION". Debajo de los campos hay dos botones: "Actualizar" (verde) y "Cancelar" (amarillo).

Finalmente, dentro de las opciones presentes en el servicio, veremos una opción de búsqueda, que nos permitirá filtrar los registros en base a un atributo en cuestión. En nuestro caso decidimos utilizar como filtro el nombre del servicio.

Representa una funcionalidad muy útil al momento de contar con gran cantidad de registro.



Interfaz de servicios del taller. El título es "SERVICIOS TALLER". Hay una barra de navegación superior con los siguientes ítems: Vehículo, Marca, Modelo, Clientes, Técnicos, Servicios, Orden Trabajo. Hay una barra de búsqueda "Buscar nombre servicio" con un icono de lupa. Hay una sección "NUEVO SERVICIO" con campos "Nombre" y "Descripción Servicio" y un botón "Registrar Servicio". Hay una sección "SERVICIOS REGISTRADOS" con una tabla de registros.

ID	Nombre Servicio	Descripción		
5	CAMBIO ACEITE	DESCRIPCION		
13	MANTENIMIENTO MOTOR	DESCRIPCION		

## **Funcionalidad - Orden Trabajo**

Por otro lado si accedemos a la opción de “Orden de trabajo” obtendremos las mismas funcionalidades descritas en los prototipos de la funcionalidad “Servicio”

En este caso a la hora de cargar los datos para una nueva orden, simplemente seleccionaremos los distintos vehículos cargados en el taller y los distintos servicios que se ofrece el mismo. Esto nos permitirá que podamos recuperar los datos del cliente que estarán relacionados al vehículo en cuestión y nos ahorrará la tarea de cargar datos de un cliente que “ya conocemos”

Nº Orden	Vehiculo	Marca	Modelo	Año	Servicio	Cliente	Fecha de Creación	
22	AIBINH-Mod	Honda	XR150	2015	MANTENIMIENTO MOTOR	Facundo Sanchez	2023-10-09	
30	AIBINH-Mod	Honda	XR150	2015	CAMBIO ACEITE	Facundo Sanchez	2023-10-09	

De igual forma se actualizará la tabla que refleja las distintas Órdenes de trabajo realizadas en el taller, la cual incluirá toda la información del cliente relacionado con el vehículo al cual se le aplica dicha orden.

Nº Orden	Vehiculo	Marca	Modelo	Año	Servicio	Cliente	Fecha de Creación	
22	AIBINH-Mod	Honda	XR150	2015	MANTENIMIENTO MOTOR	Facundo Sanchez	2023-10-09	
30	AIBINH-Mod	Honda	XR150	2015	CAMBIO ACEITE	Facundo Sanchez	2023-10-09	

Contaremos con las opciones de editar y eliminar correspondientemente, las cuales funcionan de la misma manera que hemos explicado anteriormente.

N° Orden	Vehiculo	Marca	Modelo	Año	Servicio	Cliente	Fecha de Creación
22	A181NHMad	Honda	XR150	2015	MANTENIMIENTO MOTOR	Facundo Sanchez	2023-10-09
30	A181NHMad	Honda	XR150	2015	CAMBIO ACEITE	Facundo Sanchez	2023-10-09

localhost:8080 dice

Estas seguro de eliminar?

Aceptar Cancelar

EDITAR ORDEN

A181NHMad

MANTENIMIENTO MOTOR

Actualizar Cancelar

Finalmente contamos con la opción de filtrar las ordenes de trabajo realizadas. En esta ocasión utilizamos como parámetro de filtración la fecha de las distintas ordenes de trabajo que queremos consultar.

Nº Orden	Vehiculo	Marca	Modelo	Año	Servicio	Cliente	Fecha de Creación		
22	AIBINIMod	Honda	XR150	2015	MANTENIMIENTO MOTOR	Facundo Sanchez	2023-10-09		
30	AIBINIMod	Honda	XR150	2015	CAMBIO ACEITE	Facundo Sanchez	2023-10-09		

## Organización Tareas - Segunda Versión

Para esta segunda versión del MVP hemos decidido implementar una herramienta más, llamada "Trello" la cual nos permite realizar un tablero similar al de SCRUM donde hemos podido organizar todas las tareas que tuvimos que realizar para cumplir con la entrega en cuestión.

Lista de tareas	En proceso	Hecho
Validaciones Front-End	Implementar ABM "OrdenTrabajo"	Crear Entidad "Orden Trabajo"
Validaciones Back-end	Implementar Historial de Servicios para el cliente	Crear Entidad "ServiciosTaller"
Desarrollo Documentacion		Implementar ABM "ServiciosTaller"
		Definir relaciones necesarias en la DB con las nuevas entidades
		Crear prototipo pantallas



## Consideraciones MVPs

### MVP V2

Para esta segunda versión MVP para el software de “Taller Mecánico”, se tomaron las siguientes consideraciones en base a las US propuestas por la cátedra.

- El criterio de aceptación que indica “*Los datos del cliente deben estar almacenados de manera segura y protegidos contra accesos no autorizados*” lo consideramos como una US aparte por el grado de complejidad que presenta, con lo cual no ha sido implementado para esta versión y será considerado para versiones futuras.
- El criterio de aceptación que indica “*Como usuario, quiero tener la opción de adjuntar imágenes o archivos relacionados con la orden de trabajo para un mejor seguimiento y referencia visual*” lo consideramos como una US aparte por el grado de complejidad que presenta, con lo cual no ha sido implementado para esta versión y será considerado para versiones futuras.
- Consideramos que cada servicio aplicado a un vehículo representa una orden de trabajo distinta. Con lo cual, un vehículo al cual se le realicen dos servicios distintos, le corresponderá dos órdenes de trabajo distintas.

## Documentación desarrollo - 3er Entrega

### Requerimientos propuestos

#### **Caso de Uso – Recuperar registros**

Como usuario quiero poder recuperar los datos de las marcas que han sido eliminadas, para poder restablecer los registros que necesite nuevamente.

#### **Criterios de aceptación**

- Debo poder acceder a un apartado particular donde se mostrará los registros eliminados y recuperar el que sea necesario.
- Se debe poder eliminar la totalidad de registros eliminados en caso que sea necesario para vaciar el historial de los mismos.
- Se debe poder filtrar los registros eliminados por el nombre de marca y por fecha de eliminación.

#### **Caso de Uso – Estadísticas**

Como usuario quiero poder observar estadísticas relacionadas a las órdenes de trabajo (tiempo promedio de cumplimiento de OT) y a los servicios (porcentaje de utilización), para poder tener un acceso rápido a la información.

### Criterios de aceptación

- Los datos deben poder visualizarse de forma gráfica, en un apartado particular.
- Se debe poder filtrar un periodo de tiempo específico para representar en la estadística
- Los gráficos deben poder ser descargados en formato PDF.

### **Caso de Uso – Estado OT**

Como usuario quiero poder mantener un seguimiento del estado de las órdenes de trabajo para poder conocer en todo momento en que etapa se encuentran las mismas.

### Criterios de aceptación

- A la hora de crear una nueva orden de trabajo el estado por defecto debe ser “En espera”.
- Se debe poder actualizar el estado de forma manual, entre las distintas opciones:
  - En espera
  - En progreso
  - Esperando repuestos
  - Completado.
- Se deben actualizar los estados de forma manual en el momento que sea necesario.

### **Caso de Uso – Detalle Orden Trabajo**

Como usuario quiero poder descargar los detalles de cada origen de trabajo para poder almacenar los mismos y mantener un seguimiento de las órdenes que han sido atendidas.

### Criterios de aceptación

- Los detalles descargados deben almacenarse en formato “.PDF”
- Una vez que se haya descargado un detalle, debe bloquearse el mismo para no poder seguir siendo editado.

## Postman

En esta entrega hemos implementado una funcionalidad específica que nos permite hacer utilización de la herramienta Postman. En nuestro caso tuvimos que desarrollar un controlador REST específico para que el sistema funcionase como una API y dicho programa pueda hacer uso de los end-points creados.

En esta ocasión hemos implementado un ABM para la entidad “Cliente” creando los respectivos end-points para poder hacer uso de dicha funcionalidad.

### End-Point 1

El primer endpoint me permite listar y ver todos los clientes que se encuentran almacenados dentro de la base de datos y que el backend me devuelve.

```
1 @GetMapping("/api/clientes/listar")
2     public List<Cliente> clientesPostman(){
3
4         List<Cliente> context = new ArrayList<>();
5         context = servicesCliente.listarClientes();
6         return context;
7     }
```


Usamos Postman para consumir el endpoint y ver los datos en formato JSON.

The screenshot shows the Postman interface. At the top, the method is set to 'GET' and the URL is 'http://localhost:8080/api/clientes/listar'. Below this, the 'Body' tab is selected, and the response is displayed in 'Pretty' JSON format. The response is an array of two client objects. The first object has the following details: id\_cliente: 2, nombre: 'JUAN', apellido: 'VANZETTI', dni: 43883934, telefono: 3534297983, localidad: 'VILLA MARIA', nroLicencia: '3', and vtoLicencia: '2023-12-22'. The second object starts with id\_cliente: 4, nombre: 'PEDRO', and apellido: 'MARTINEZ'.

```
1 [
2   {
3     "id_cliente": 2,
4     "nombre": "JUAN",
5     "apellido": "VANZETTI",
6     "dni": 43883934,
7     "telefono": 3534297983,
8     "localidad": "VILLA MARIA",
9     "nroLicencia": "3",
10    "vtoLicencia": "2023-12-22"
11  },
12  {
13    "id_cliente": 4,
14    "nombre": "PEDRO",
15    "apellido": "MARTINEZ",
```

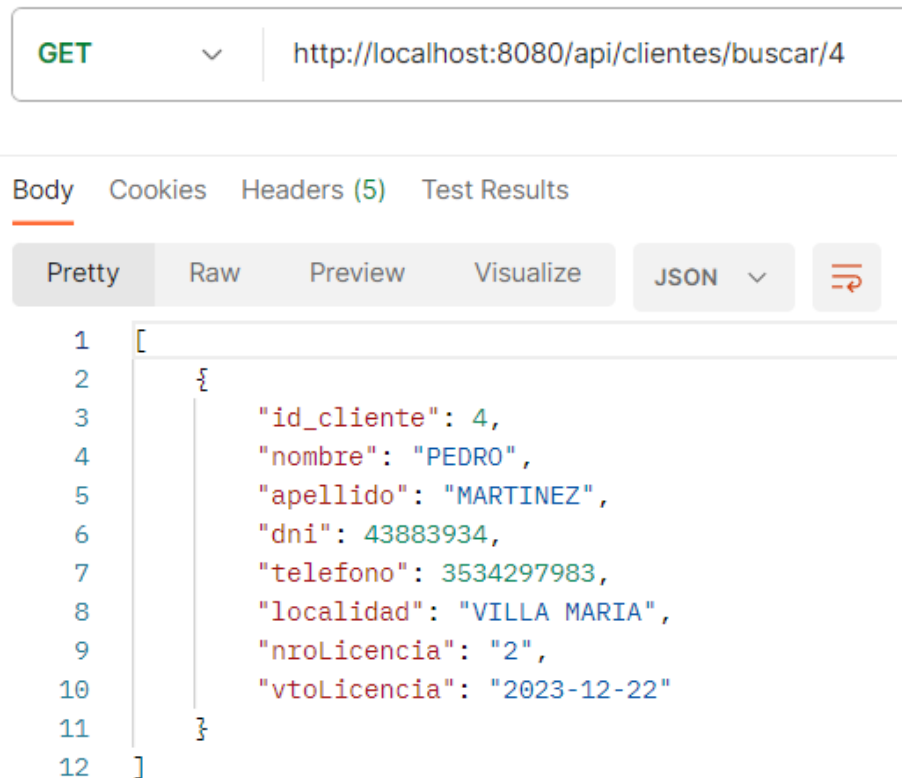
## End-Point 2

El siguiente endpoint permite buscar un cliente en específico mediante el ID que pasamos como parámetro por la url. Para ello debemos conocer dicho ID, en caso contrario devolverá una lista vacía.



```
1  @GetMapping("/api/clientes/buscar/{id_buscar}")
2  public List<Cliente> clientesPostmanID(@PathVariable int id_buscar){
3
4      List<Cliente> clientes = servicesCliente.listarClientes();
5      List<Cliente> clientesBuscados = new ArrayList<>();
6
7      for(Cliente c:clientes){
8          if (c.getId_cliente().equals(id_buscar)){
9              clientesBuscados.add(c);
10         }
11     }
12
13     return clientesBuscados;
14
15
16 }
```

En este paso pasamos el ID = 4 mediante Postman para buscar los datos de dicho cliente.

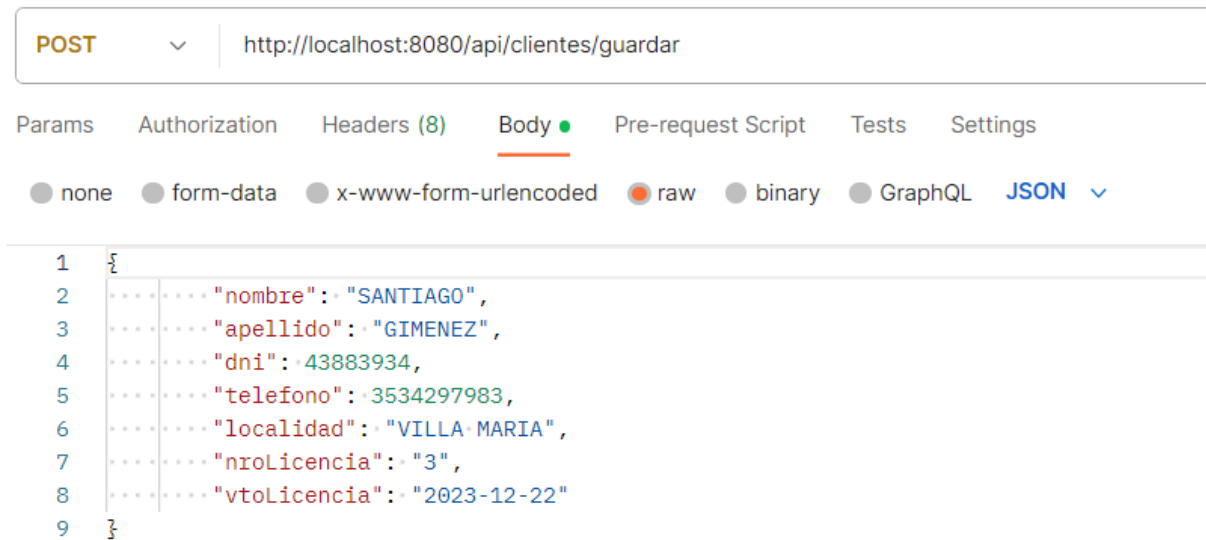


### End-Point 3

La tercera opción es poder crear un nuevo objeto "Cliente" mediante un método POST al backend y pasándole a través el body un JSON para poder hacer esto.



En Postman nos aseguramos que al momento de construir el body, cumplamos con cada uno de los atributos definido en la entidad en cuestión, este caso en “Cliente”, así como también el tipo de dato que acepta cada uno de las mismas.

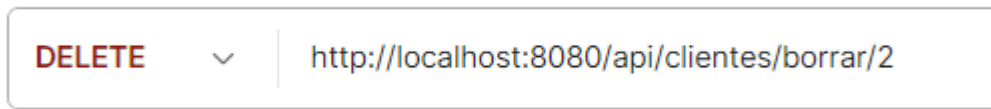


## End-Point 4

Este endpoint nos permite eliminar un cliente en particular mediante el ID del mismo. Con lo cual es necesario, nuevamente, conocer dicho ID para llevar a cabo esta operación.



En postman únicamente llevamos a cabo la petición “DELETE” al endpoint correspondiente, pasando el ID del objeto cliente que queremos eliminar de la base de datos y automáticamente es eliminado.

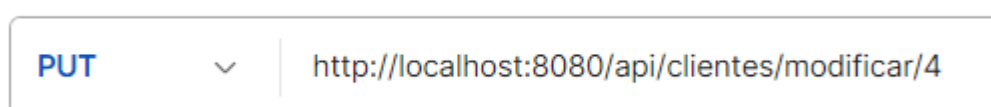


## End-Point 5

La última opción disponible es modificar los datos de algún cliente en particular. Para ello es necesario pasar la ID del cliente el cual va a ser modificado y mediante el body la información actualizada, de esta forma, se modificarán los datos dentro de la base de datos en el objeto correspondiente.



En Postman enviamos el ID del cliente mediante URL y a través del body la información actualizada.



Objeto Cliente con ID = 4 antes de ser actualizado:

```
{
  "id_cliente": 4,
  "nombre": "FACUNDO",
  "apellido": "GRIBAUDO",
  "dni": 43883934,
  "telefono": 3534297983,
  "localidad": "VILLA MARIA",
  "nroLicencia": "2",
  "vtoLicencia": "2023-12-22"
},
```

Body enviado:

```
{
  "id_cliente": 4,
  "nombre": "PEDRO",
  "apellido": "MARTINEZ",
  "dni": 43883934,
  "telefono": 3534297983,
  "localidad": "VILLA MARIA",
  "nroLicencia": "2",
  "vtoLicencia": "2023-12-22"
}
```

Objeto actualizado:

```
{
  "id_cliente": 4,
  "nombre": "PEDRO",
  "apellido": "MARTINEZ",
  "dni": 43883934,
  "telefono": 3534297983,
  "localidad": "VILLA MARIA",
  "nroLicencia": "2",
  "vtoLicencia": "2023-12-22"
},
}
```

Esta es la funcionalidad aplicada para hacer uso de la herramienta Postman. Puede llevarse a cabo la misma lógica en otras entidades para variar.



## Codacy

Otra de las herramientas que hemos implementado en esta versión final del sistema “Taller Mecanico” consiste en “Codacy”, esta herramienta es online y nos ayuda a verificar aspectos de calidad sobre nuestro código. Evalúa buenas prácticas así como también los errores que cometemos y cualquier otro aspecto que podríamos mejorar.

Para comenzar a hacer uso de esta herramienta nos dirigimos a su [sitio web](#) y nos logueamos en la misma.

Una vez dentro, vinculamos nuestra cuenta con “GitHub” en caso de que hayamos logueado con otro usuario. Esto nos permitirá administrar y evaluar nuestros repositorios.

A continuación añadimos el repositorio que queremos evaluar, en nuestro caso “TallerMecanicoPA2023”.

### Manage repositories

[Take a tour](#) [About this page](#) [×](#)

Add or follow your GitHub repositories to start analysing them. Make sure that Codacy has permissions to access your repositories.  
To add repositories, you need to have Admin permissions on them.

[Missing some repositories?](#)

Repository name	Last updated	Actions
<b>TallerMecanicoPA2023</b>	an hour ago	<a href="#">Go to repository</a>

Y autorizamos la evaluación del mismo. Una vez finalice podremos ver los resultados en el panel que ofrece Codacy.

En nuestro caso analizamos la rama “TEST” de nuestro repositorio, sobre la cual estábamos desarrollando y obtuvimos los siguientes resultados:



## Issues breakdown



Podemos observar que tenemos únicamente errores de tres tipos

- Code Style
- Error Prone
- Security

A la hora de analizar los mismos en detalle pudimos observar que la mayoría consisten en errores de malas prácticas lo cual hace que seamos redundantes en nuestro código o dupliquemos el mismo.

Fuera de esto, no tenemos errores tan significativos.

## Jconsole

Utilizamos la herramienta de JCONSOLE para poder ver el rendimiento de nuestro equipo a medida que el sistema se encuentra en funcionamiento. Esto nos permite determinar los requerimientos mínimos de sistema que necesitamos para utilizar el software, así como también para evaluar si un equipo con el cual ya contamos, es capaz de hacer uso de dicho sistema o no.

Jconsole ya **viene incluido dentro del JDK de Java**, con lo cual lo único que tenemos que hacer para utilizarlo es una serie de pasos para configurar el sistema.

Los pasos en cuestión son los siguientes:

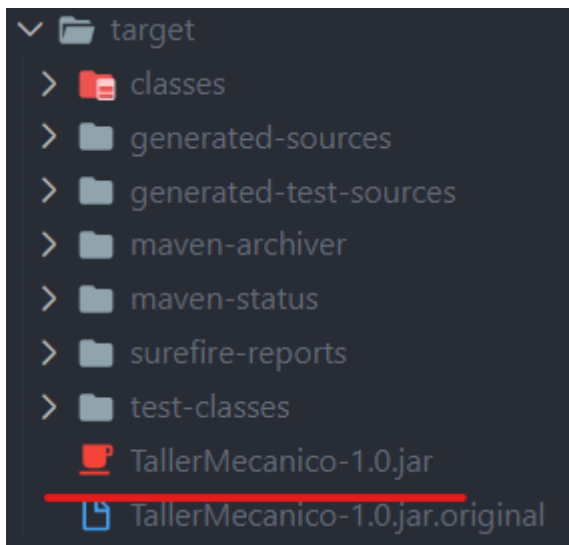
- Verificamos que tengamos instalado el JDK de Java.
- Verificamos que tenemos las variables de entorno de nuestro sistema correctamente configuradas para hacer uso Jconsole. Nos aseguramos que la variable de entorno **“Path”** de nuestro equipo apunta a la carpeta **“Bin”** del JDK que tengamos instalado.

Por otro lado creamos una nueva variable de entorno denominada “**JAVA\_HOME**” que apunta al directorio raíz de nuestro JDK.

- Configuramos el archivo main de nuestro proyecto de spring boot. Agregamos el decorador **@EnableMBeanExport** e importamos el módulo necesario para su correcto funcionamiento.

```
1  package com.Proyecto.TallerMecanico;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  import org.springframework.context.annotation.EnableMBeanExport;
7
8  @SpringBootApplication
9  @EnableMBeanExport
10 public class TallerMecanicoApplication {
11
12     public static void main(String[] args) {
13         SpringApplication.run(TallerMecanicoApplication.class, args);
14     }
15
16 }
```

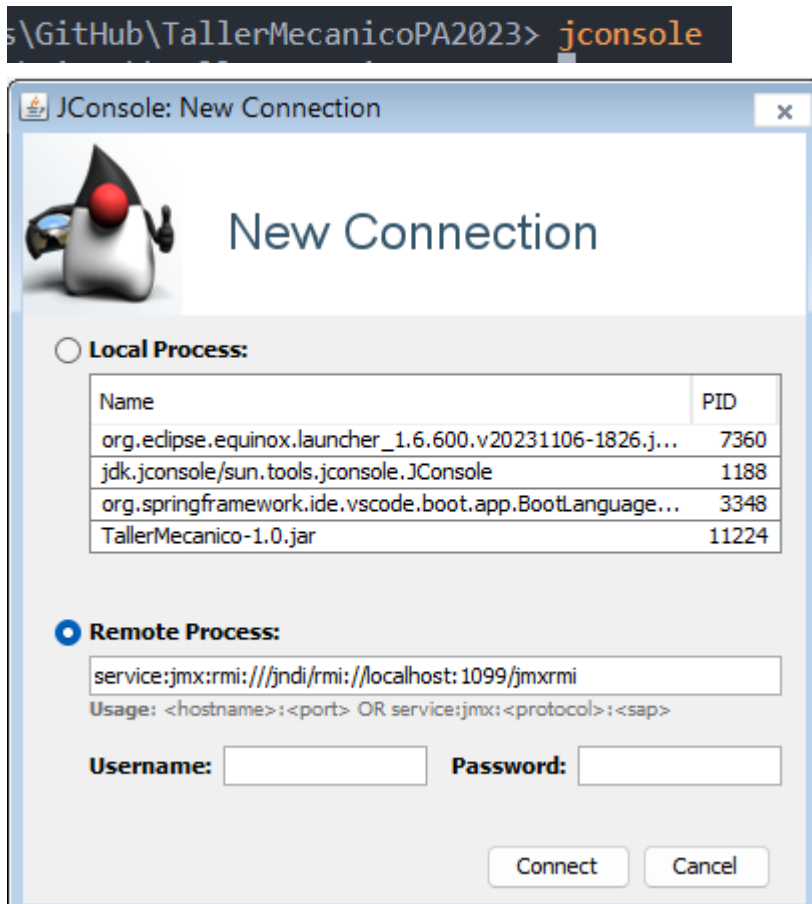
- Abrimos una nueva terminal y nos dirigimos al directorio “Target” por defecto, que es donde se encuentra el archivo compilado de nuestro sistema (.jar). En caso de que dicho archivo este en otro directorio, nos dirigimos a él.



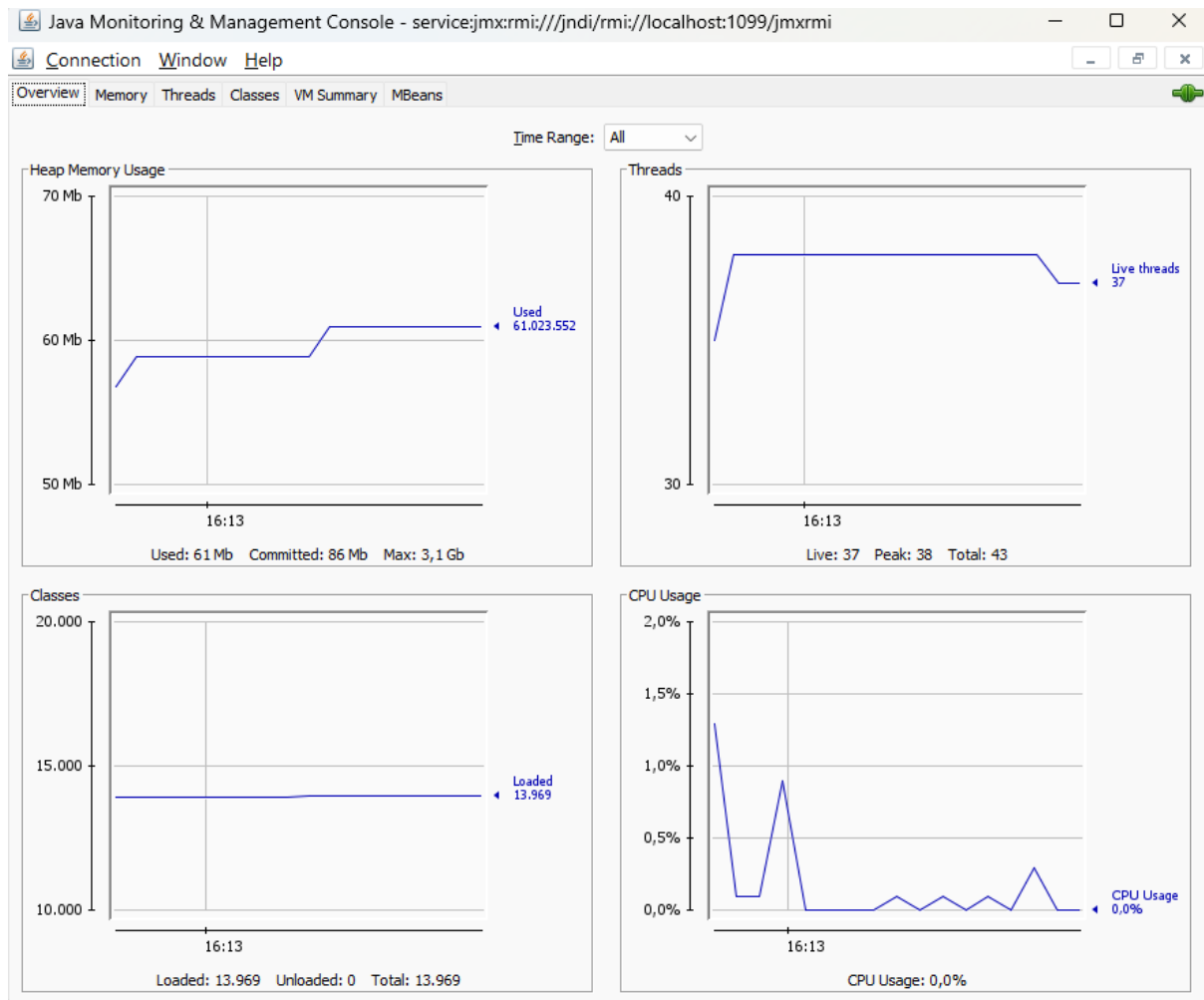
- Dentro de dicho directorio, ejecutamos el sistema pasándole una serie de configuraciones que nos permitirán hacer uso de Jconsole.

```
java -Dcom.sun.management.jmxremote `
    -Dcom.sun.management.jmxremote.port=1099 `
    -Dcom.sun.management.jmxremote.ssl=false `
    -Dcom.sun.management.jmxremote.authenticate=false `
    -jar 'TallerMecanico-1.0.jar'
```

- Ejecutó “Jconsole” en la terminal para abrir la interfaz gráfica que provee y poder conectarnos de manera **remota** al servidor local que hemos configurado al iniciar el sistema mediante el siguiente link (service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi)



Una vez realizados estos pasos podremos comenzar a ver gráficamente el rendimiento de nuestro equipo a la hora de utilizar el sistema.




En nuestro caso en particular el sistema no consume casi recursos con lo cual puede ser utilizado en cualquier equipo, por más que sea de bajos recursos.

## JUnit


JUnit es una dependencia que nos permite llevar a cabo testing sobre nuestro código, definiendo una serie de pruebas unitarias para validar que se cumplen los distintos criterios de aceptación.

Para implementar JUnit lo primero que debemos realizar es instalar las dependencias necesarias en nuestro archivo pom.xml



```
1 <dependency>
2   <groupId>org.junit.jupiter</groupId>
3   <artifactId>junit-jupiter-api</artifactId>
4   <version>5.10.1</version>
5 </dependency>
```

Luego debemos crear una clase de configuración que nos permita implementar las distintas pruebas unitarias que queramos.



```
1 package com.Proyecto.TallerMecanico;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.boot.test.context.SpringBootTest;
5
6 @SpringBootTest
7 class TallerMecanicoApplicationTests {
8
9     @Test
10     void contextLoads() {
11     }
12
13 }
```

Finalmente creamos las clases que definirán los test que queramos realizar. En nuestro caso hemos implementado pruebas unitarias para dos entidades distintas, “Cliente” y “Modelo”.

## Test - Cliente

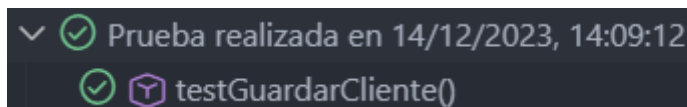
Para la entidad cliente hemos definido un único test el cual valida que sea posible crear distintos objetos clientes y poder almacenarlos dentro de la base de datos. Esta validación nos permite asegurarnos que podemos trabajar con distintos clientes dentro del sistema y administrar los mismos.

```

1  @SpringBootTest(classes = TallerMecanicoApplication.class)
2  @ComponentScan(basePackages = "com.Proyecto.TallerMecanico")
3  public class ClienteTest {
4
5      @Autowired
6      private ClienteServices testCliente;
7
8      /*
9       * TEST1 PARA VERIFICAR QUE ES POSIBLE GUARDAR OBJETOS "CLIENTE"
10     */
11     @Test
12     public void testGuardarCliente(){
13
14         //Cliente Prueba
15         Cliente c = new Cliente("Cliente", "TEST", BigInteger.valueOf(43883934), BigInteger.valueOf(43883934), "deheza", "1", "2");
16
17         //Evaluo test
18         Integer resultado = testCliente.save(c);
19         Assertions.assertEquals(1, resultado);
20
21     }
22 }

```

Ejecutamos el test definido y nos aseguramos de que pasa las pruebas, esto nos valida el test.



✓ Prueba realizada en 14/12/2023, 14:09:12  
 ✓ testGuardarCliente()

## Test - Modelo

Para la entidad "Modelo" hemos definido dos test distintos.

- Test 1 → Nos permite verificar que podamos crear objetos modelo y almacenarlos en la base de datos. Es la misma lógica que el test sobre la entidad "Cliente".
- Test 2 → Nos permite verificar que cada modelo existente dentro de nuestro sistema se encuentra asociado a un objeto de la entidad "Marca". Con este test nos aseguramos que cada modelo está asociado a una marca en cuestión y nos permite asegurarnos que cumplimos con la regla de negocio que indica esta cuestión, ya que no pueden existir modelos "aislados" es decir, que no estén relacionados a marcas existentes.



```
1  /*
2      * TEST1 - PARA VERIFICAR QUE ES POSIBLE GUARDAR OBJETOS "MODELO"
3      */
4  @Test
5  public void testGuardarModelo(){
6
7      //Objeto marca para crear el modelo de prueba
8      Marca mar = new Marca("MarcaPrueba", "Activo");
9      //Modelo
10     Modelo m = new Modelo("Modelo1", "Test", mar);
11
12     //Evaluo test
13     Integer resultado = testModelo.save(m);
14     Assertions.assertEquals(1, resultado);
15
16 }
```

✓ ✓ Prueba realizada en 14/12/2023, 14:10:37

✓ ✓ testGuardarModelo()



```
1  /*
2      * TEST2 - PARA VERIFICAR QUE CADA MODELO PERTENECE A UNA MARCA
3      */
4      @Test
5      public void testMarcaAsociada(){
6
7          List<Modelo> modelos = testModelo.listarModelos();
8
9          Boolean resultado = false;
10
11         for (Modelo m:modelos){
12             if (m.getMarca().getNombre() != ""){
13                 resultado = true;
14             }
15         }
16
17         Assertions.assertEquals(true, resultado);
18     }
19
```

✓ ✓ Prueba realizada en 9/12/2023, 15:39:22  
✓ ✓ testMarcaAsociada()

Estas son las pruebas unitarias que hemos implementado para poder hacer utilización de la herramienta Junit.