

**PROFESOR:**

JUAN JOSE VANZETTI

**GRupo 13 - INTEGRANTES:**

EXEQUIEL SUAREZ

Mateo gerbaudo

agustin bartoloni

**VERSION:**

VERSION 4.0 (Entrega Final)

proyecto integrador programacion avanzada

auto s.a.

INDICE

[Introducción 2](#_Toc162731031)

[Requisitos Funcionales 2](#_Toc162731032)

[Repositorio 3](#_Toc162731033)

[Requisitos del Sistema 3](#_Toc162731034)

[Configuración de la Base de Datos 4](#_Toc162731035)

[Estructura del Proyecto 5](#_Toc162731036)

[Diagrama de Entidad-Relación (DER) 6](#_Toc162731037)

[Configuración de la Base de Datos y Creación Automática de Tablas 6](#_Toc162731038)

[Dependencias de Maven 6](#_Toc162731039)

[Ejecución del Proyecto 8](#_Toc162731040)

[Pruebas 8](#_Toc162731041)

[Despliegue 8](#_Toc162731042)

[Estadísticas 9](#_Toc162731043)

[Monitoreo de Consumo de Memoria con Jconsole 10](#_Toc162731044)

[Pruebas Unitarias 12](#_Toc162731045)

Documentación del Proyecto

# Introducción

Este documento proporciona información esencial sobre el proyecto Spring Boot desarrollado por [Nombre del Proyecto]. Este proyecto utiliza Java como lenguaje de programación principal, junto con el framework de persistencia Hibernate, una base de datos MySQL para almacenar datos y Maven para la gestión de dependencias.

Desarrollado por los integrantes del grupo 13, de la catedra Programación avanzada, en la Facultar Regional de Villa María.

# Requisitos Funcionales

Orden de Trabajo:

* **Crear Orden de Trabajo:** Se puede crear una nueva orden de trabajo proporcionando información como cliente, vehículo asociado, técnico, servicio y observaciones.
* **Listar Ordenes de Trabajo:** El sistema permite obtener la lista de todas las órdenes de trabajo creadas.
* **Filtrar Ordenes de Trabajo:** Las órdenes de trabajo se pueden filtrar por estado (pendientes, anuladas, finalizadas).
* **Eliminar Orden de Trabajo:** Se puede eliminar una orden de trabajo existente.
* **Ver Detalle de Orden de Trabajo:** Se puede ver el detalle de una orden de trabajo, incluyendo información sobre quién es el encargado de realizarla y los servicios asociados.
* **Acciones disponibles en la Orden de Trabajo:** NroOrden, Patente, Marca, Modelo, Fecha, Estado, Acciones.

Administración de Datos:

* **Técnico:**
  + Se puede crear, buscar, editar y eliminar técnicos. También se puede filtrar por nombre y DNI.
  + Campos de un técnico: Legajo, DNI, Apellido, Nombre, Teléfono, Email, Domicilio, Acciones.
* **Servicio:**
  + Se puede crear, editar y eliminar servicios. Los servicios tienen una descripción, precio sin impuesto y acciones disponibles.
* **Cliente:**
  + Se puede agregar, buscar, editar y eliminar clientes. También se puede buscar por DNI, ID y nombre.
  + Campos de un cliente: ID, DNI, Apellido, Nombre, Teléfono, Email, Domicilio, Acciones.
* **Vehículo:**
  + Se puede agregar, buscar, editar y eliminar vehículos. Se pueden filtrar por patente y cliente asociado.
  + Campos de un vehículo: Patente, Marca, Modelo, Año, Kilometraje, Acciones.
* **Modelo:**
  + Se puede agregar, buscar, editar y eliminar modelos. También se puede buscar por marcas.
  + Campos de un modelo: ID, Nombre, Marca, Acciones.
* **Marcas:**
  + Se pueden agregar marcas con un impuesto fijo del 15%. Se pueden buscar todas sin filtros.
  + Campos de una marca: ID, Nombre, Impuesto, Acciones.
* **Factura:**
  + Se pueden visualizar e imprimir las facturas facturadas. Se puede filtrar por ID

# Repositorio

[**https://github.com/gerbaudo19/Auto-S.A-Testbed**](https://github.com/gerbaudo19/Auto-S.A-Testbed)

# Requisitos del Sistema

BackEnd:

* **Java JDK 17:** Versión de Java utilizada para el desarrollo del BackEnd y la ejecución de pruebas.
* **Maven 3.1.3:** Herramienta de gestión de proyectos utilizada para construir y gestionar las dependencias del BackEnd.
* **Sistema Operativo:** Windows, macOS o Linux.
* **Servidor MySQL:** Instalado y configurado para la persistencia de datos en el BackEnd.
* **Hibernate, JPA:** Tecnologías utilizadas para la capa de persistencia en el BackEnd.
* **XAMPP:** Se recomienda instalar XAMPP para crear un entorno de desarrollo local completo que incluye Apache (para el servidor web), MySQL (para la base de datos) y facilita la configuración del entorno.

Pruebas del BackEnd:

* **JUnit 5:** Marco de pruebas unitarias utilizado para asegurar la corrección de las unidades individuales del código del BackEnd.
* **Mockito:** Utilizado para crear objetos simulados (mocks) y realizar pruebas de comportamiento en entornos controlados.

Herramientas de Testing del BackEnd:

* **Postman:** Herramienta utilizada para la prueba de las funcionalidades del BackEnd. Permite realizar pruebas de integración y verificar el correcto funcionamiento de las API.

FrontEnd:

* **HTML5, CSS, Bootstrap 5, JavaScript:** Tecnologías utilizadas en el FrontEnd para el desarrollo de las interfaces de usuario y la lógica del cliente.

# Configuración de la Base de Datos

Texto

Descripción generada automáticamente

# Estructura del Proyecto

El proyecto sigue una estructura de paquetes y módulos estándar de Spring Boot. Aquí está la estructura principal del proyecto:

Texto

Descripción generada automáticamente

# 

# Diagrama de Entidad-Relación (DER)

Diagrama

Descripción generada automáticamente

# Configuración de la Base de Datos y Creación Automática de Tablas

Para la gestión de la base de datos en este proyecto, se utiliza Hibernate en conjunto con JPA (Java Persistence API). Esto permite que las tablas de la base de datos se generan automáticamente en función de las clases de entidad de Java que hemos definido en el código. Solo se realiza la creación de la base de datos con el Gestor de base de datos MySQL para su conexión.

Dependencias de Maven

El archivo `pom.xml` contiene las dependencias de Maven utilizadas en el proyecto. Asegúrate de que estas dependencias estén correctamente configuradas y actualizadas.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# Ejecución del Proyecto

Para compilar y ejecutar el proyecto localmente, sigue estos pasos:

# Clonar el repositorio del proyecto.

git clone https://github.com/gerbaudo19/Auto-S.A-Testbed.git

# Navegar al directorio raíz del proyecto.

cd Auto-S.A-Testbed

# Ejecutar el siguiente comando Maven para compilar y ejecutar la aplicación:

mvn spring-boot:run

# Pruebas

Para ejecutar las pruebas de integración que involucran la base de datos MySQL, asegúrate de tener una base de datos de prueba configurada correctamente y modifica la configuración de prueba en `application.properties`.

# Despliegue

Para desplegar la aplicación en un entorno de producción, sigue las prácticas recomendadas de empaquetado y despliegue de Spring Boot. Asegúrate de configurar la base de datos de producción según las necesidades específicas.

Esta documentación proporciona una visión general del proyecto Spring Boot y su configuración básica. Asegúrate de mantenerla actualizada a medida que el proyecto evoluciona y cambia.

# 

# Estadísticas

*Todas las estadísticas tienen un periodo anual.*

**1- Servicios Más Populares:**

**Definición de Estadística:** Lista de los tres servicios más solicitados en un periodo de tiempo.

**Explicación:** Identificar los servicios más populares ayuda a adaptar la oferta del taller y a centrarse en las áreas de mayor demanda. Esta estadística permite a los propietarios y administradores del taller tomar decisiones informadas sobre la asignación de recursos y la promoción de servicios específicos.

**2- Comparativa de Rendimiento entre Técnicos:**

**Definición de Estadística:** Clasificación de técnicos según el número total de servicios completados en un periodo de tiempo.

**Explicación:** Proporciona una visión rápida del rendimiento relativo entre técnicos, sin detallar demasiado, facilitando la identificación de áreas de mejora. Esta estadística puede utilizarse para reconocer a los técnicos más productivos, así como para identificar a aquellos que podrían necesitar capacitación adicional o apoyo.

**3- Ganancia Total:**

**Definición de Estadística:** Ingreso total en un periodo de tiempo.

**Explicación:** Esta estadística proporcionará una visión general del rendimiento financiero del taller, permitiendo a los usuarios evaluar la salud financiera del negocio. Conocer la ganancia total en un periodo de tiempo determinado es fundamental para la toma de decisiones financieras, como la planificación presupuestaria, la evaluación de la rentabilidad y la identificación de áreas de mejora.

**4- Estadística de Estado de Órdenes:**

**Definición:**

La estadística de estado de órdenes presenta la siguiente información:

* **Pendiente:** Representa el número de órdenes de trabajo que están en espera de ser procesadas o completadas.
* **Anulado:** Indica la cantidad de órdenes de trabajo que han sido canceladas o anuladas antes de su finalización.
* **Finalizado:** Muestra el total de órdenes de trabajo que han sido completadas y no requieren más acciones.

**Explicación:** Esta estadística proporciona una visión general de la distribución de las órdenes de trabajo según su estado actual. Permite a los usuarios entender rápidamente la distribución de las órdenes de trabajo en diferentes etapas de su ciclo de vida, lo que puede ser útil para la planificación y la toma de decisiones. Además, ayuda a identificar posibles cuellos de botella en el proceso de trabajo y a optimizar la eficiencia operativa del taller.

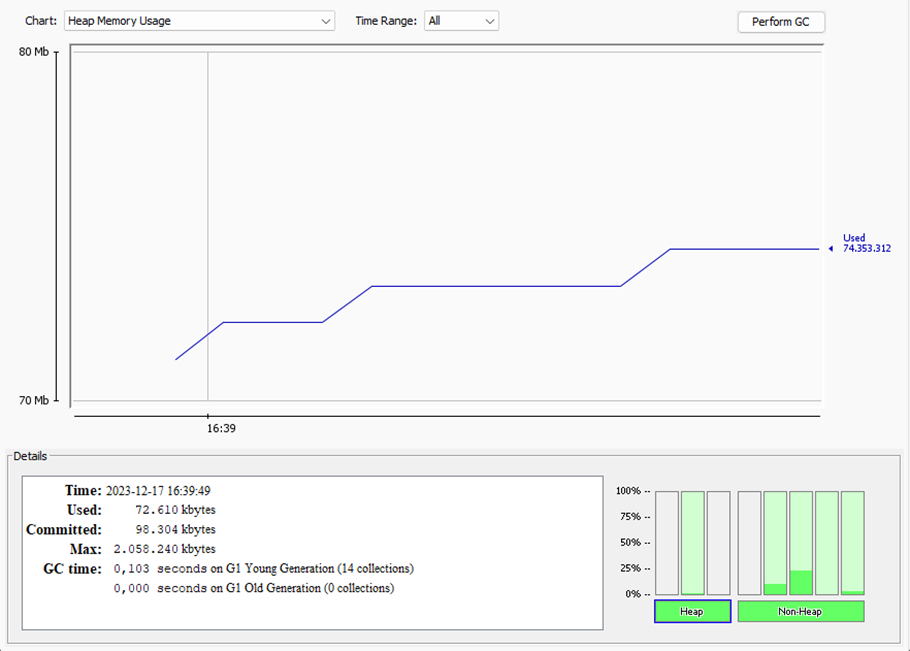




# Monitoreo de Consumo de Memoria con Jconsole

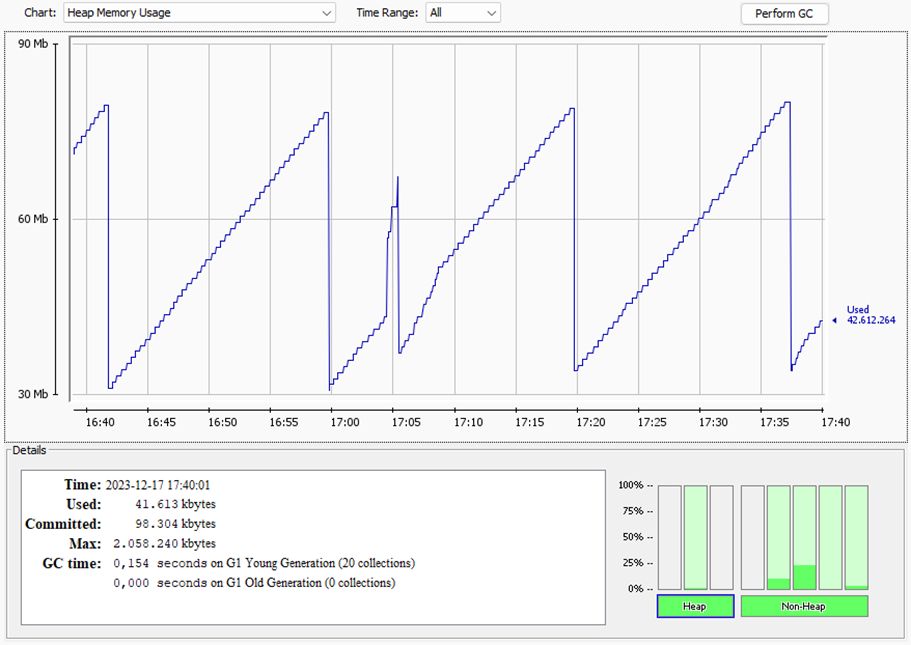
Captura 1: Estado Inicial del Consumo de Memoria

La primera captura revela el estado del consumo de memoria en el momento inicial de la ejecución del sistema. Proporciona una visión instantánea de la asignación de memoria y sirve como referencia para evaluar cualquier cambio significativo.



Captura 2: Consumo de Memoria Después de una Hora

La segunda captura muestra el estado del consumo de memoria después de una hora de funcionamiento continuo. Se ha observado que el sistema mantiene una estabilidad en el uso de la memoria durante este período.



Este monitoreo detallado del consumo de memoria a lo largo del tiempo es fundamental para evaluar la eficiencia y la estabilidad del sistema, proporcionando insights valiosos sobre posibles problemas de gestión de memoria y permitiendo la optimización continua del rendimiento.

# Pruebas Unitarias

Las pruebas unitarias son una parte fundamental del desarrollo de software en nuestro proyecto. Permiten verificar el correcto funcionamiento de unidades individuales de código, como métodos y funciones, de manera aislada y repetible. Aquí se describen las prácticas y convenciones que seguimos al escribir y ejecutar pruebas unitarias en nuestro proyecto.

**Herramientas Utilizadas**

En nuestro proyecto, utilizamos las siguientes herramientas para escribir y ejecutar pruebas unitarias:

* **JUnit**: Framework de pruebas unitarias para Java.
* **Mockito**: Biblioteca de Mockito para crear y verificar comportamientos de objetos simulados (mocks) en las pruebas.
* **Spring Boot Test**: Integración de Spring Boot con pruebas unitarias y de integración.
* **Maven**: Herramienta de gestión de proyectos para construir y administrar dependencias, incluidas las bibliotecas de pruebas.

**Convenciones de Nomenclatura**

Seguimos las siguientes convenciones de nomenclatura al escribir pruebas unitarias:

* Los nombres de las clases de prueba deben tener el mismo nombre que la clase que están probando, seguido de "Test". Por ejemplo, MiClaseTest.
* Los métodos de prueba deben comenzar con el prefijo "test" seguido de una descripción del comportamiento que están probando.
* Se le realizo las clases de pruebas a todas las clases (entity -repository – dto – service - controller) que se encuentran en el Diagrama de Entidad-Relación.

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class MiClaseTest {

@Test

public void testSuma() {

// Arrange

MiClase miClase = new MiClase();

// Act

int resultado = miClase.suma(2, 3);

// Assert

assertEquals(5, resultado);

}

}