AlquiCar

# Plan de Pruebas y resultados

## 1. Información General

- **Nombre del Proyecto:** Alquicar
- **Propósito del Plan de Pruebas:** Verificar que las funcionalidades de las clases (`Vendedor`, `Vehiculo`, `Cliente`, `Boleta`, `ArrayVendedor`, `arreglo_Cliente`, `ArregloBoleta`, `ArregloVehiculo`) se comporten como se espera.
- **Herramientas:** JUnit 5, Eclipse/IDE.
- **Alcance de las Pruebas:**
    - Validación de métodos *getters*, *setters*, constructores y lógica de negocio.
    - Pruebas de listas dinámicas (agregar, eliminar, actualizar, buscar).
    - Carga y guardado de datos en archivos de texto.

## 2. Estrategia de Pruebas

- **Niveles de Prueba:**
    - Pruebas Unitarias: Para cada método en cada clase.
    - Prueba de funcionamiento a través de SonnarQube
- **Casos de Prueba Automatizados:**
    - Todas las pruebas se ejecutan con JUnit 5.
    - Se evalúa tanto el flujo exitoso como los escenarios negativos.

## 3. Casos de Prueba

A continuación, se describen los casos de prueba principales para las clases:

### *3.1 Clase Vendedor*

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Validar constructor con todos los campos | Datos válidos de vendedor | Objeto `Vendedor` creado correctamente |
| Probar *setters* y *getters* | Asignar valores válidos | Los valores se almacenan y recuperan correctamente |

### 3.2 Clase `Vehiculo`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Validar constructor completo | Datos válidos de vehículo | Objeto `Vehiculo` creado correctamente |
| Actualizar campo `combustible` | Cambiar a "Híbrido" | Campo actualizado sin errores |

### 3.3 Clase `Cliente`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Crear cliente con datos válidos | Datos válidos de cliente | Objeto `Cliente` creado correctamente |
| Eliminar cliente | Cliente existente | Cliente eliminado correctamente de la lista |

### 3.4 Clase `Boleta`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Adicionar boleta válida | Datos válidos de boleta | Boleta agregada a la lista |
| Buscar boleta inexistente | Código no existente | Retorna `null` |

### 3.5 Clase `ArrayVendedor`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Agregar vendedor existente | Objeto ya en la lista | Devuelve mensaje de "El vendedor ya existe" |
| Verificar cantidad total de vendedores | Lista inicial y datos nuevos | Devuelve el total actualizado |

### 3.6 Clase `arreglo_Cliente`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Actualizar datos del cliente | Cliente existente | Cliente actualizado correctamente |
| Buscar cliente por código | Código válido | Retorna el cliente correcto |

### 3.7 Clase `ArregloBoleta`

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Validar grabación en archivo | Lista con datos | Se guarda archivo `boletas.txt` correctamente |

### *3.8 Clase `ArregloVehiculo`*

| Caso de Prueba | Entrada | Resultado Esperado |
|---|---|---|
| Listar vehículos por marca | Marca "Toyota" | Devuelve lista de vehículos `Toyota` |
| Buscar vehículo no existente | Código inválido | Retorna `null` |

## 4. Criterios de Aceptación

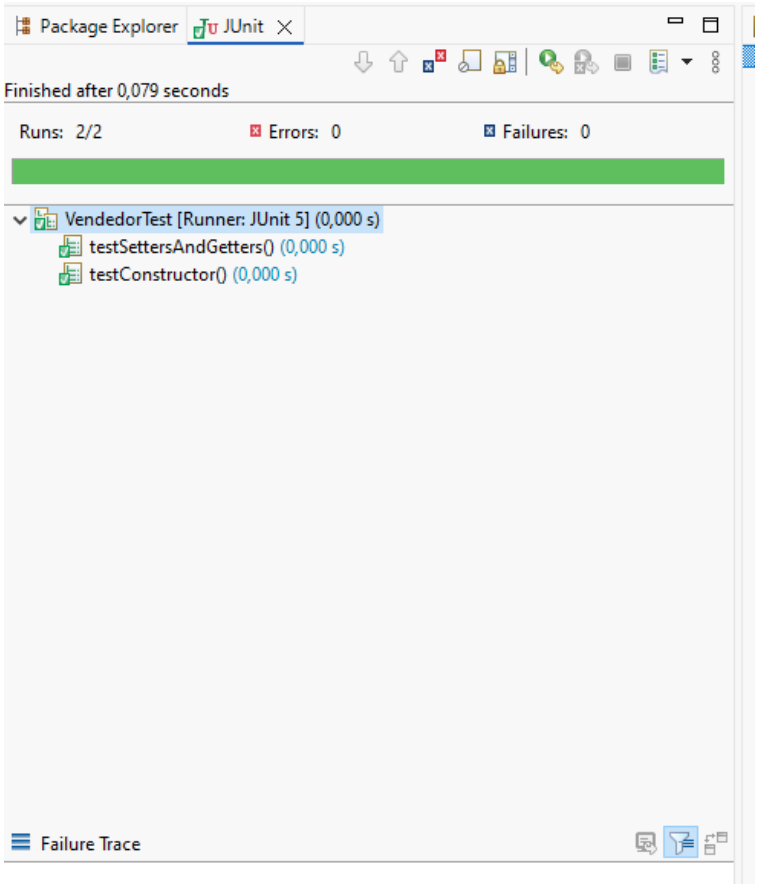
- Todas las pruebas deben ejecutarse sin errores.
- El programa tiene que pasar la prueba de SonnarQube

## 5. Herramientas Requeridas

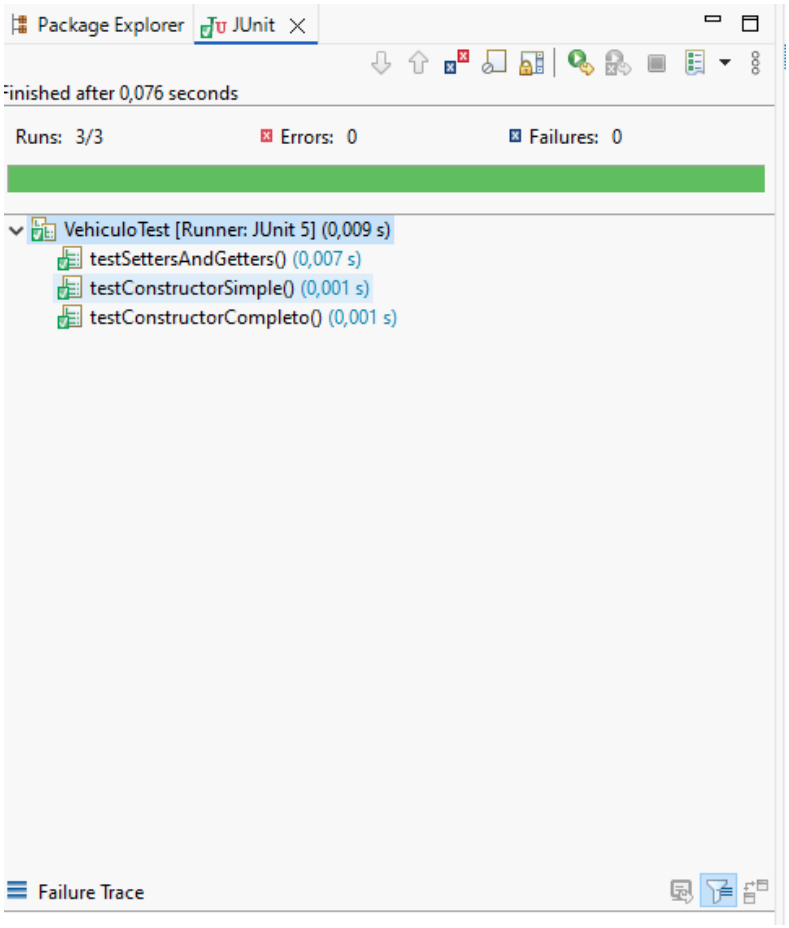
- **JUnit 5:** Para la ejecución de las pruebas.
- **Eclipse:** Para facilitar la visualización.
- **SonnarQube y SonnarScanner:** Para verificar la integridad del programa.

## 6. Resultados

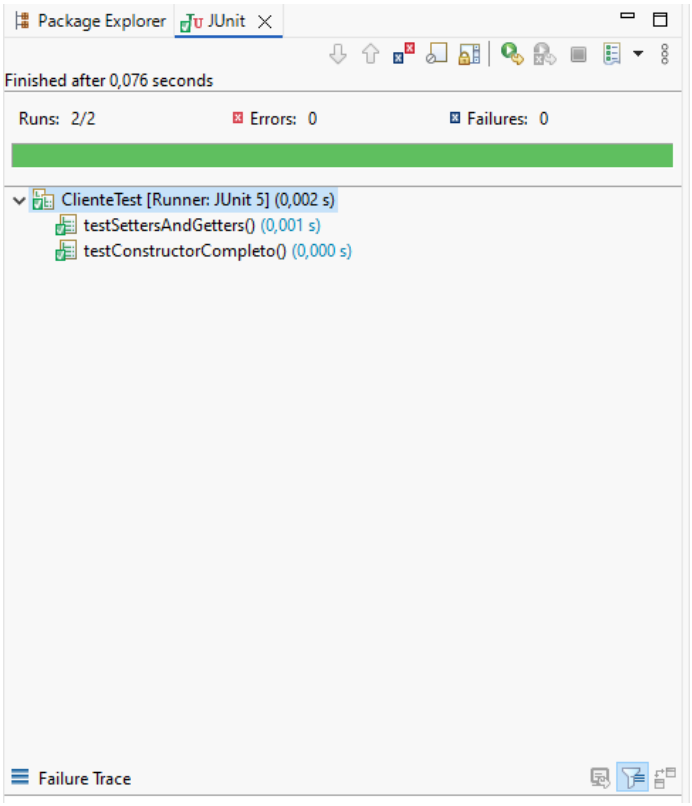
### *6.1 clase `Vendedor`*



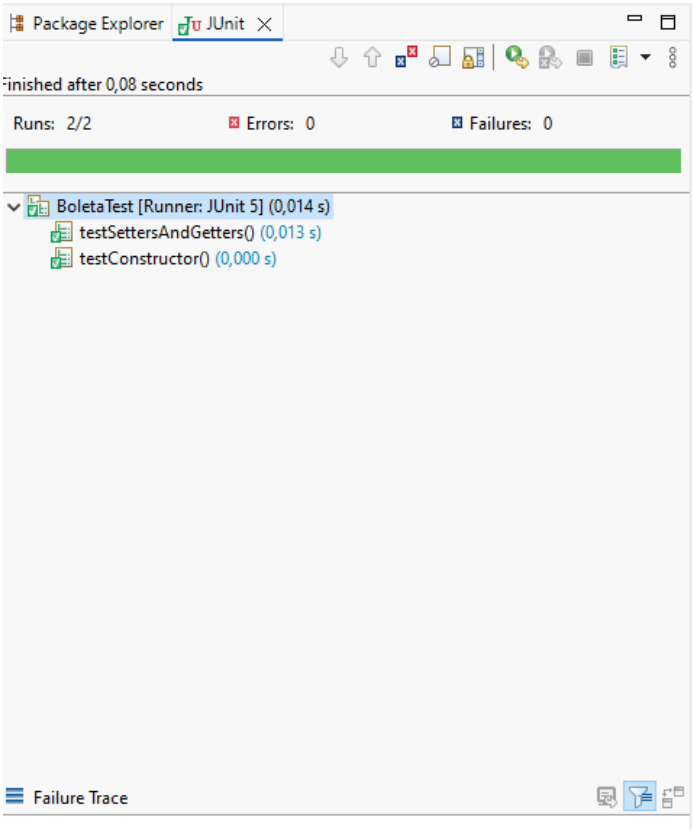
### 6.2 Clase *Vehiculo*



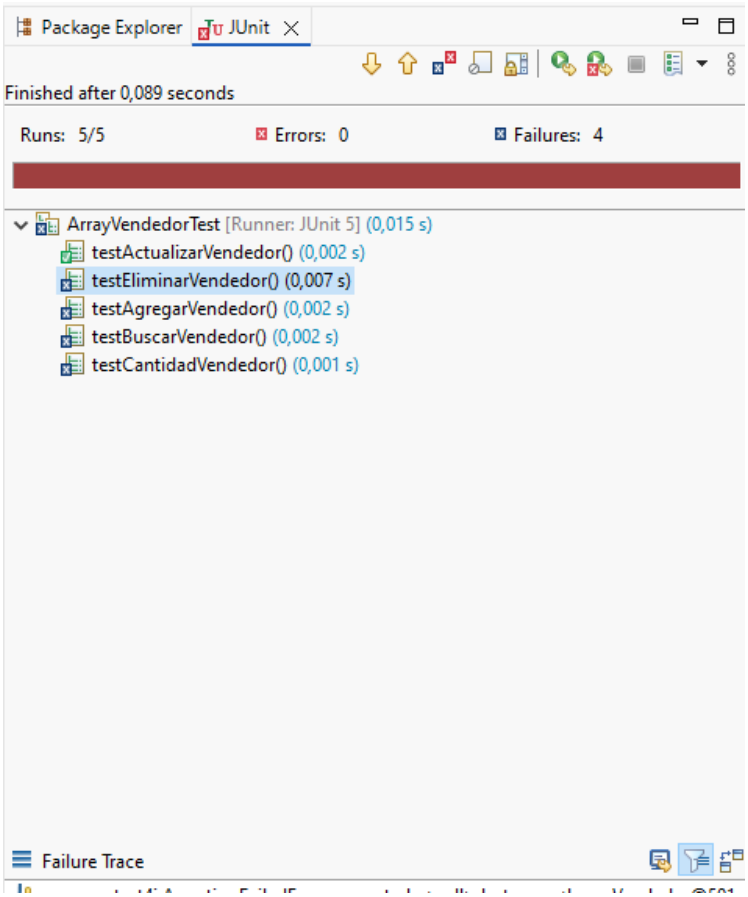
### 6.3 Clase `Cliente`



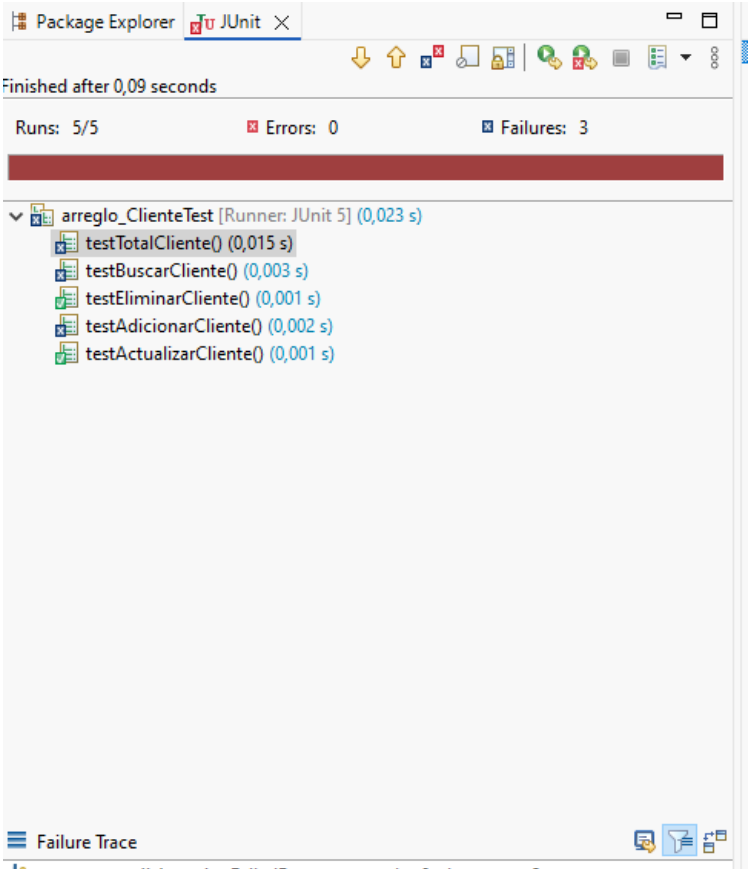

### 6.4 Clase `Boleta`



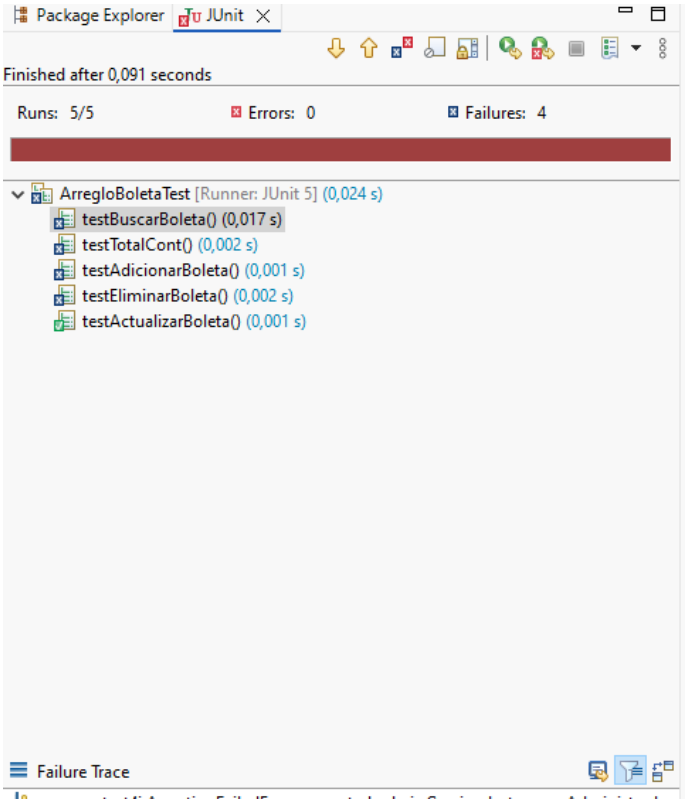
### 6.5 Clase *ArrayVendedor*



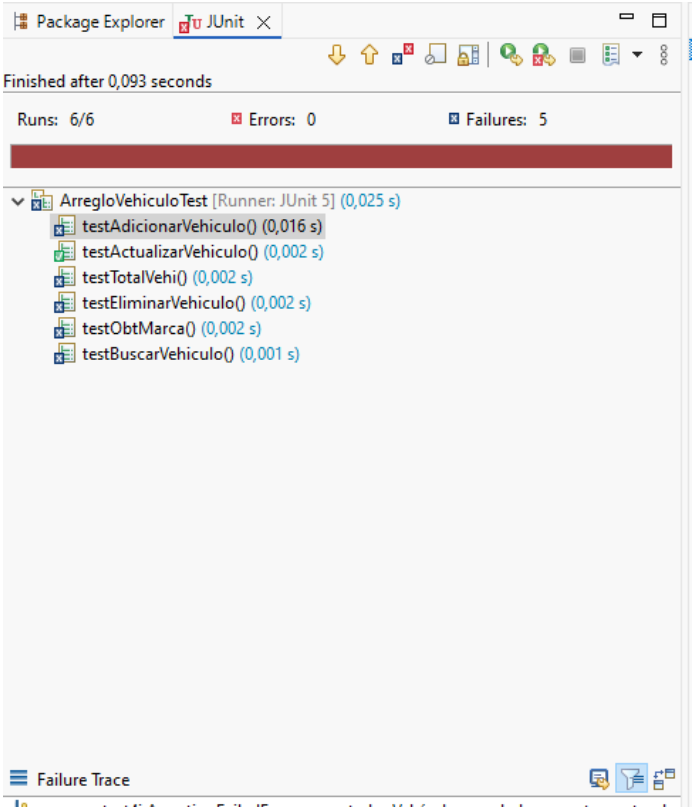

### 6.6 Clase *arreglo_Cliente*



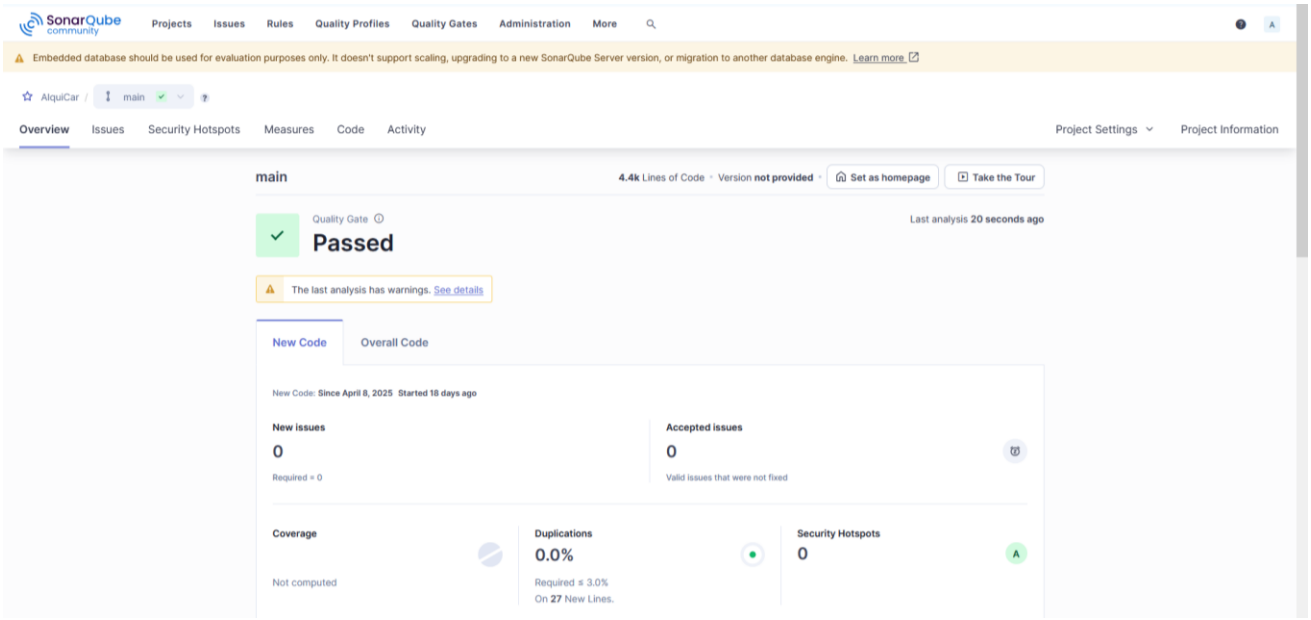
### 6.7 Clase *ArregloBoleta*




### 6.8 Clase *ArregloVehiculo*



### 6.8 Pruebas de SonnarQube




## 6. Conclusiones

El programa funciona correctamente y las pruebas no muestran fallos, no obstante, en las clases “ArrayVendedor”, “Arreglo_Cliente”, “ArregloBoleta” y “ArregloVehículo” Se han mostrado varios fallos, lo cual implica que el código funciona correctamente pero no como esperábamos en las pruebas, por lo que harbrá que continuar con las pruebas.