**RETO TÉCNICO SOFKA:**

Reto Técnico: Training League Desarrollo de un inventario de tipos de naves espaciales utilizando el paradigma POO

**DESARROLLADOR**

Daniel Vallejo Merino

**SOFKA TRAINING LEAGUE 2023**

**COLOMBIA   
10 DE FEBRERO DE 2023**

**RETO TÉCNICO:**

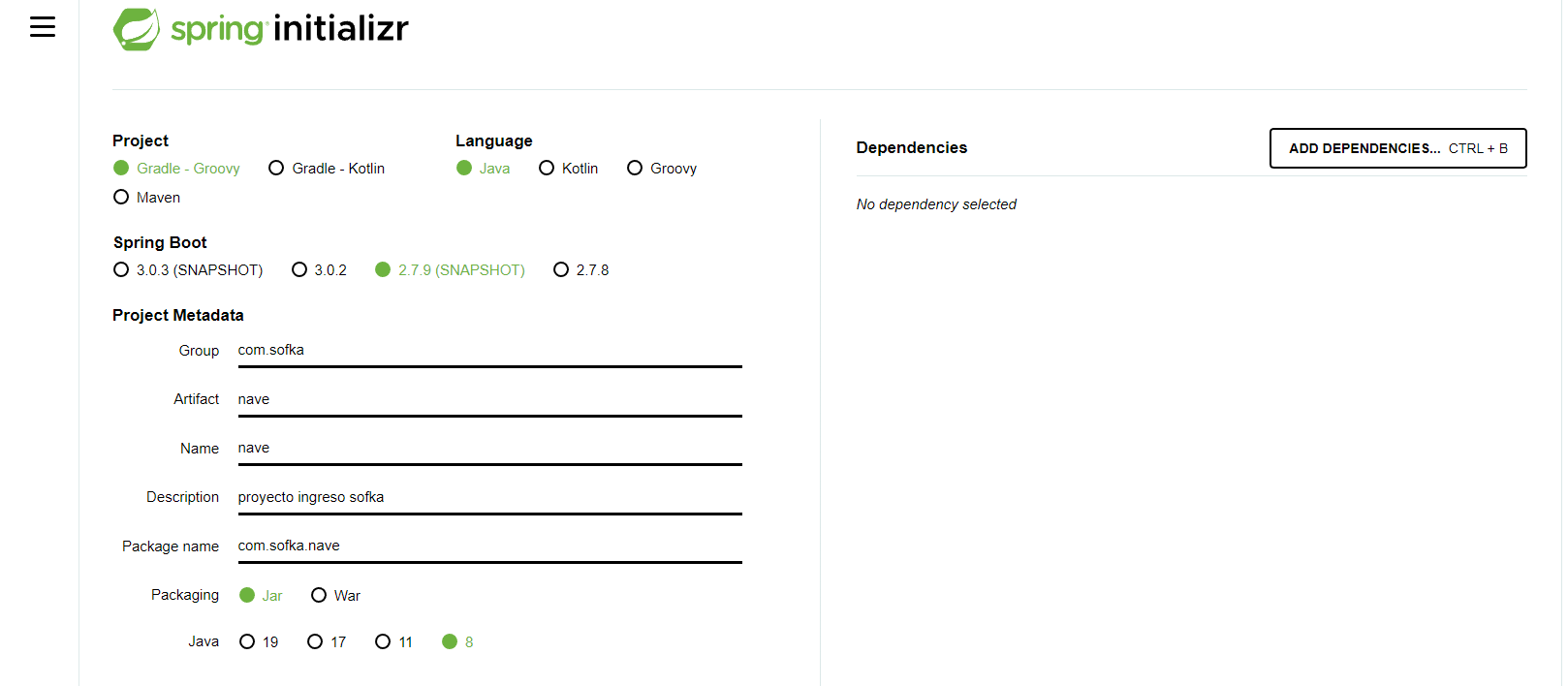
El objetivo del presente informe es documentar el código realizado para la elaboración del reto técnico de Sofka empleando el paradigma de programación orientada a objetos POO el cual consiste en el registro y consulta de diferentes tipos de naves espaciales, teniendo en cuenta la abstracción, herencia, interfaces y polimorfismo extrayendo los atributos pertinentes y las clases abstractas requeridas para el planteamiento del problema.

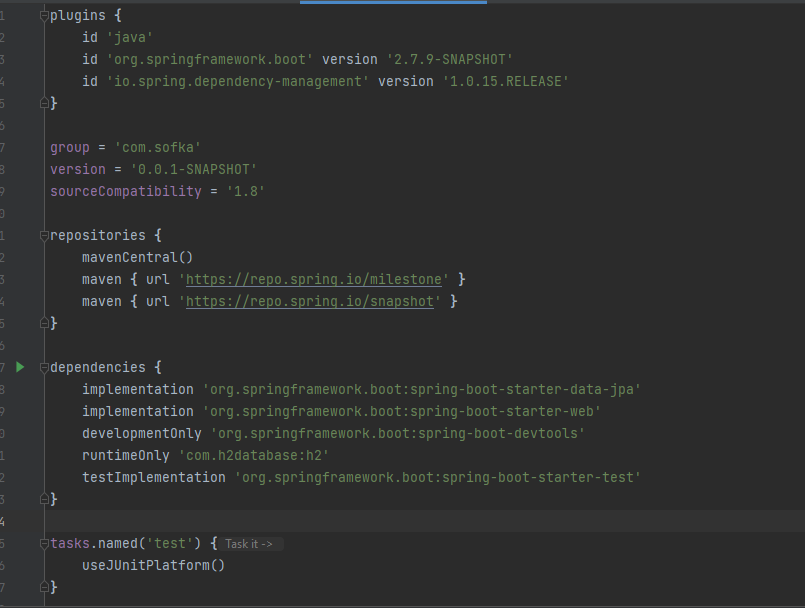
**RESUMEN**

En el presente informe se desarrolla un programa el cual permite registrar y consultar diferentes tipos de naves espaciales de acuerdo con el artículo suministrado de referencia, del cual se logran distinguir principalmente tres tipos de naves espaciales, los vehículos lanzadera, las naves espaciales no tripuladas y las tripuladas. El código del BACK-END se realizó empleando el IDE de IntelliJ IDEA, empleando el JDK 1.8.0\_351 de Oracle, adicionalmente se utilizó Spring Boot como asistente para la creación de un proyecto Gradle y posteriormente descargar las dependencias necesarias para el desarrollo del proyecto.

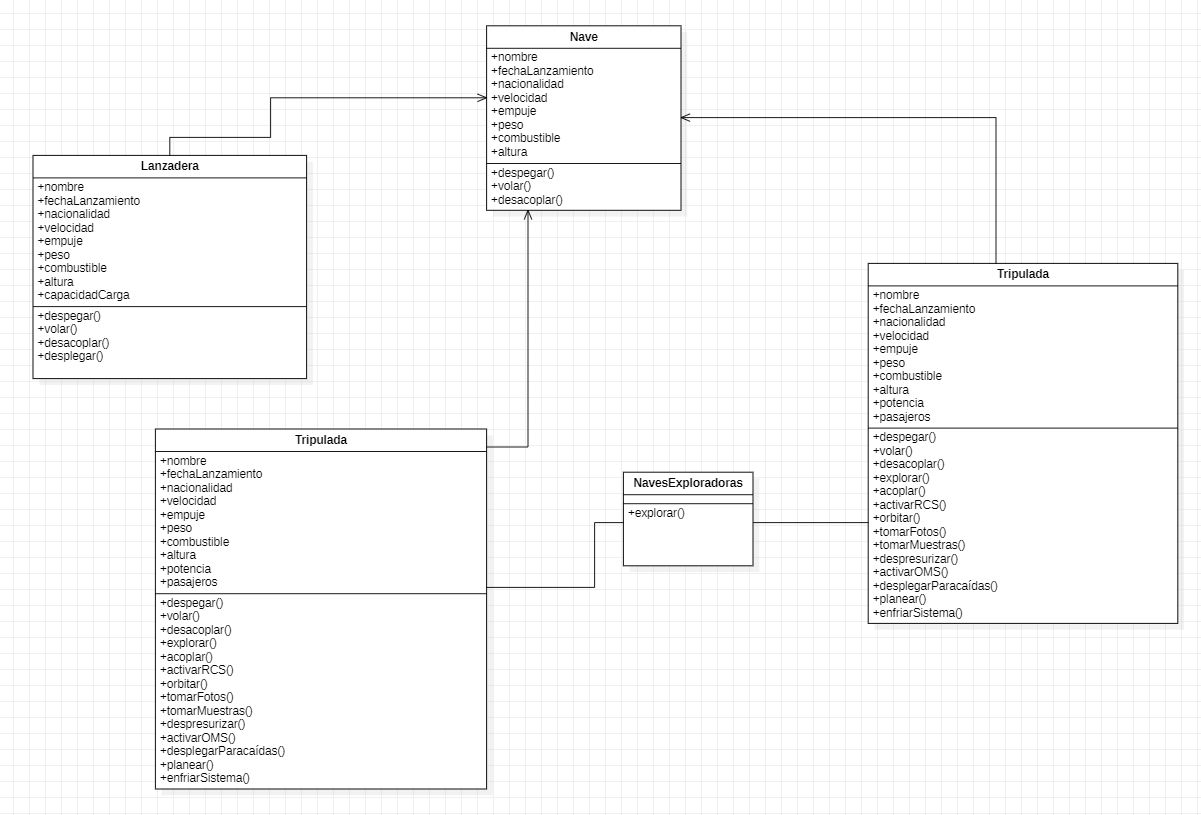
**Método y Resultados**

Como método, se empleó Spring Boot para la realizaron sesiones sincrónicas, con el objetivo de realizar el componente correspondiente a la capa de presentación de la aplicación, con el fin de permitir la interacción con el usuario final, considerando las funcionalidades ya desarrolladas en los sprints anteriores y realizadas en el componente lógico. Teniendo en cuenta lo anterior, se crearon cada una de las historias de usuario y posteriormente se asignaron las tareas correspondientes a cada uno de los desarrolladores.



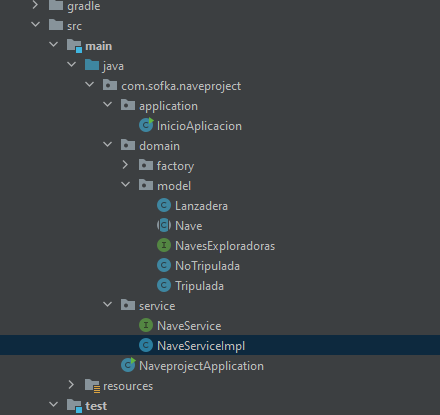


Posteriormente se realizó el diagrama de clases y atributos, para de esta forma plantear la estructura del proyecto, empleando la herencia para el desarrollo del código

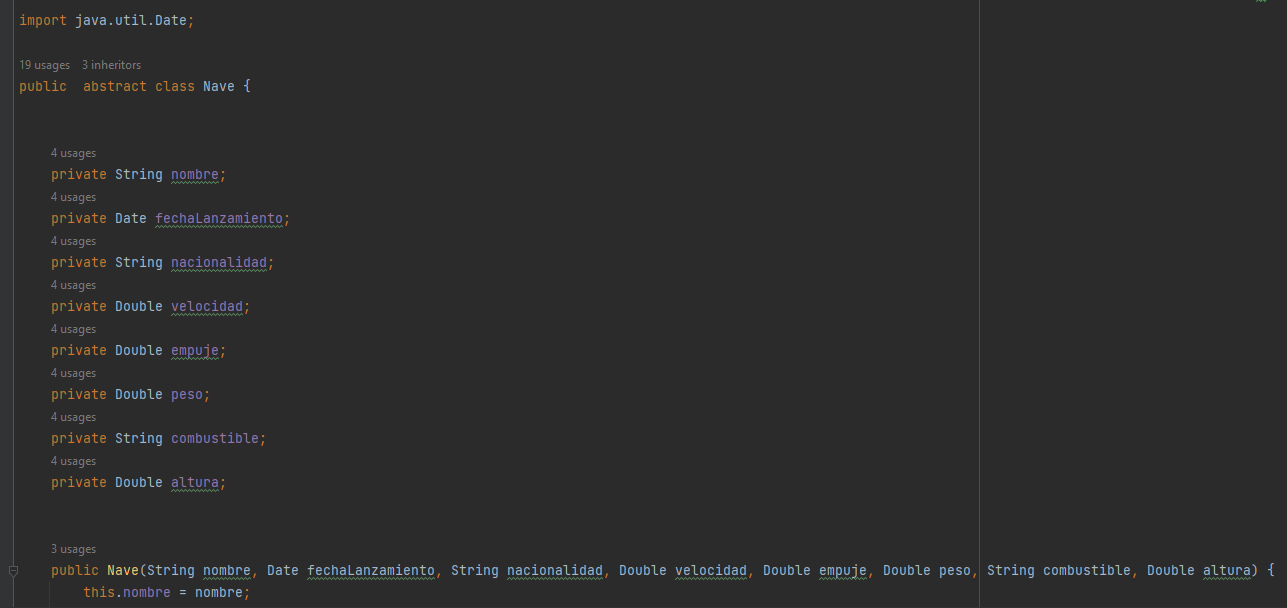


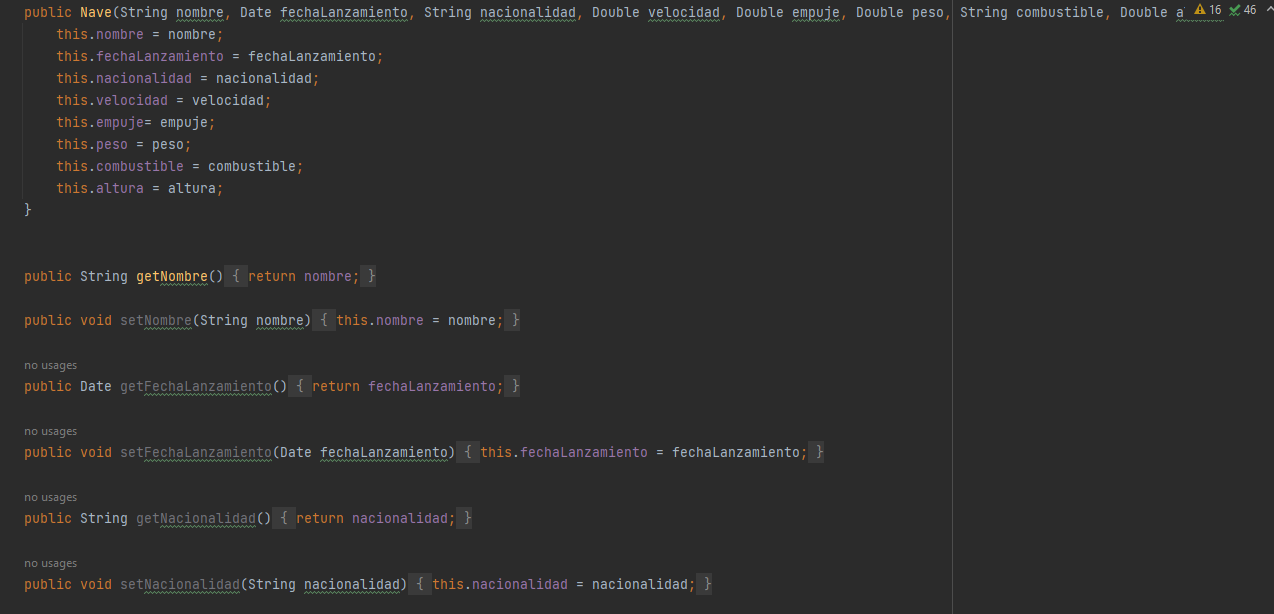
De lo anterior se puede evidenciar que se identificaron principalmente tres tipos de naves espaciales, las lanzaderas, las naves tripuladas y las no tripuladas las cuales heredan atributos de la clase padre nave y a su vez cada una de las categorías posee sus propios atributos. También se identificó una nueva categoría, las naves exploradoras las cuales se definieron como una interfaz, ya que solo las naves tripuladas y no tripuladas cuentan con esta habilidad. Los métodos también se sobrecargan a partir de la clase padre y asi mismo cada tipo de nave, posee sus propios métodos.

Se empleó el estilo de arquitectura MVC modelo vista controlador, dichas secciones se logran identificar en el IDE, donde se distingue la sección modelo donde está el código de cada una de las clases, el controlador delimitado por service trabajando como intermediario entre el modelo y la vista y la vista la cual hace alusión al Inicio de la aplicación, donde se encuentra el interfaz de usuario



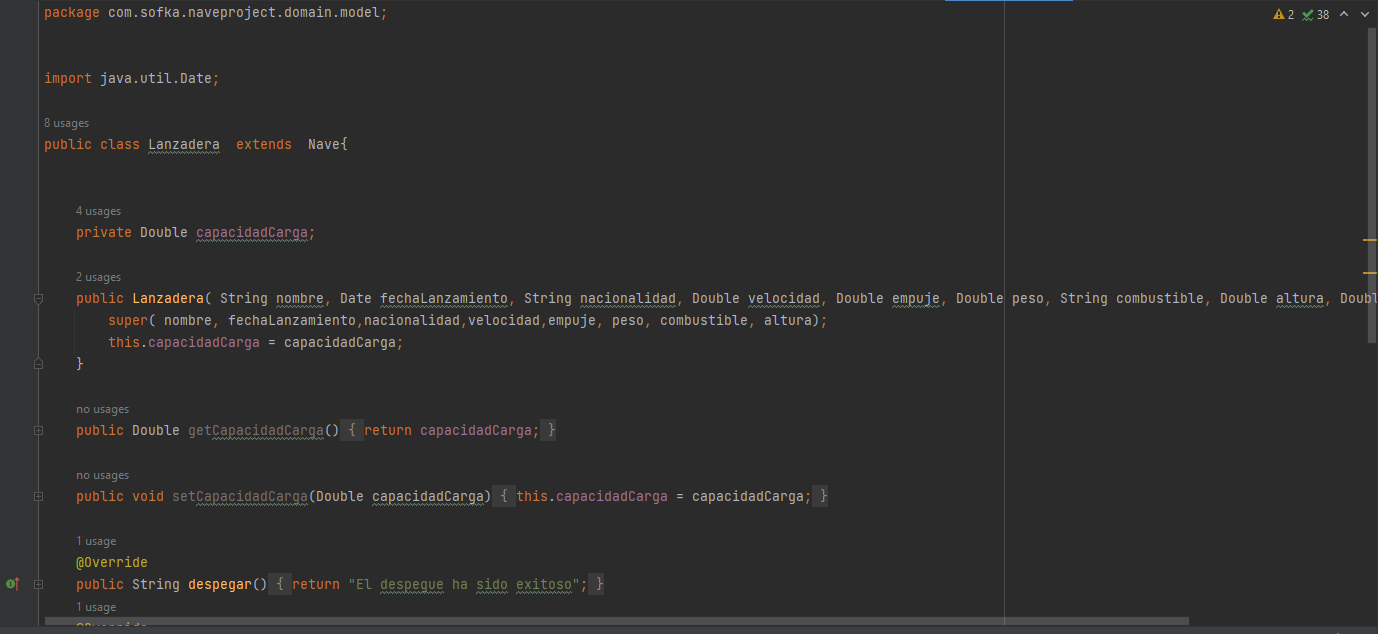
En el modelo, primero se definió la clase padre nave con sus atributos y métodos, por tal motivo se empleó como una clase abstracta, posteriormente se crearon las clases hijas Lanzadera, NoTripulada y Tripulada heredando los atributos de la clase padre y sobrecargando los métodos

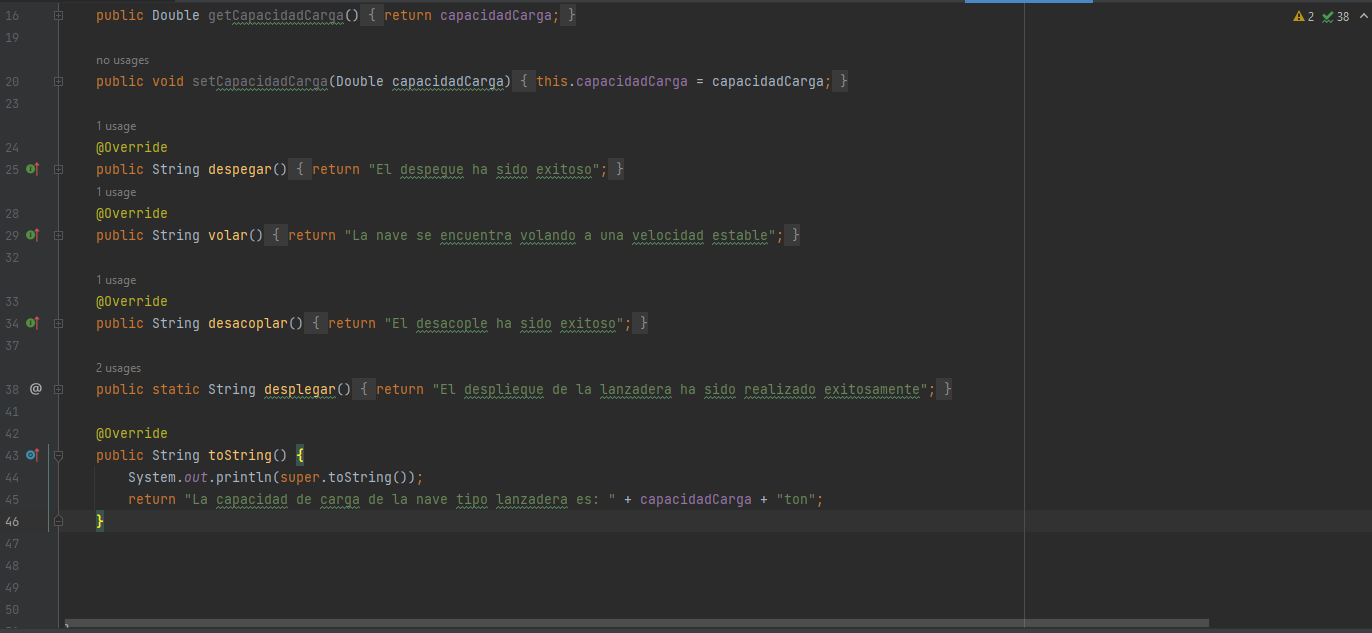




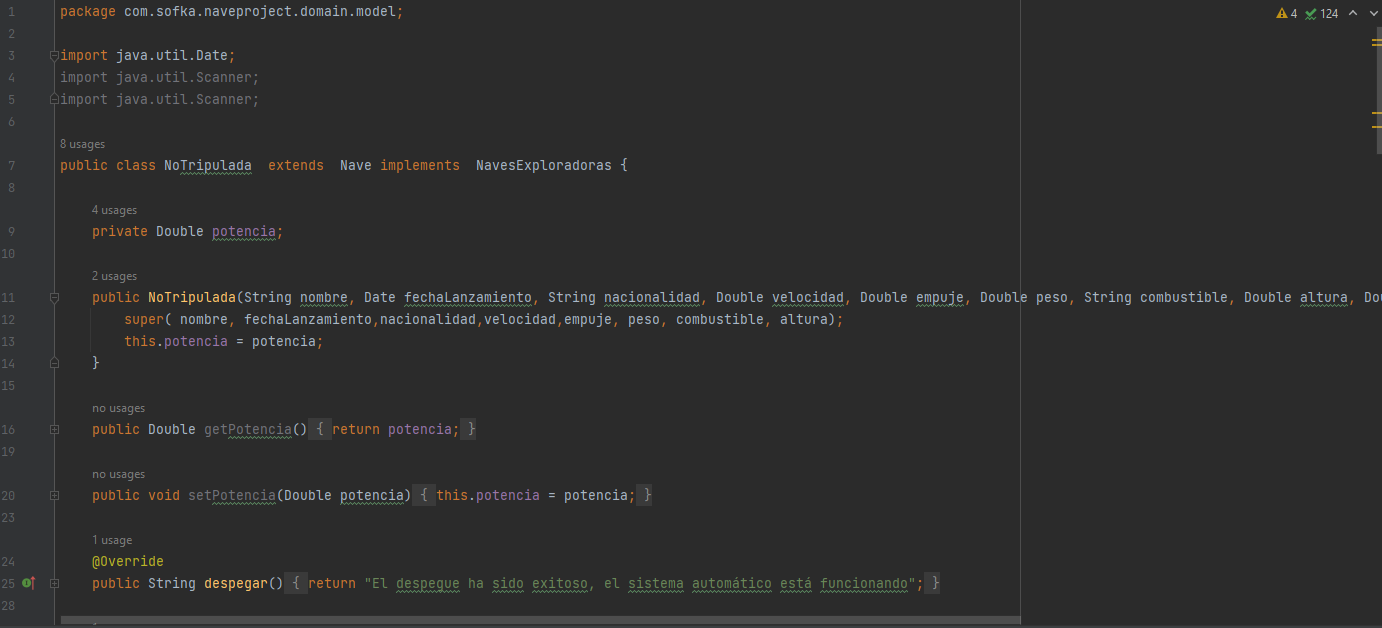


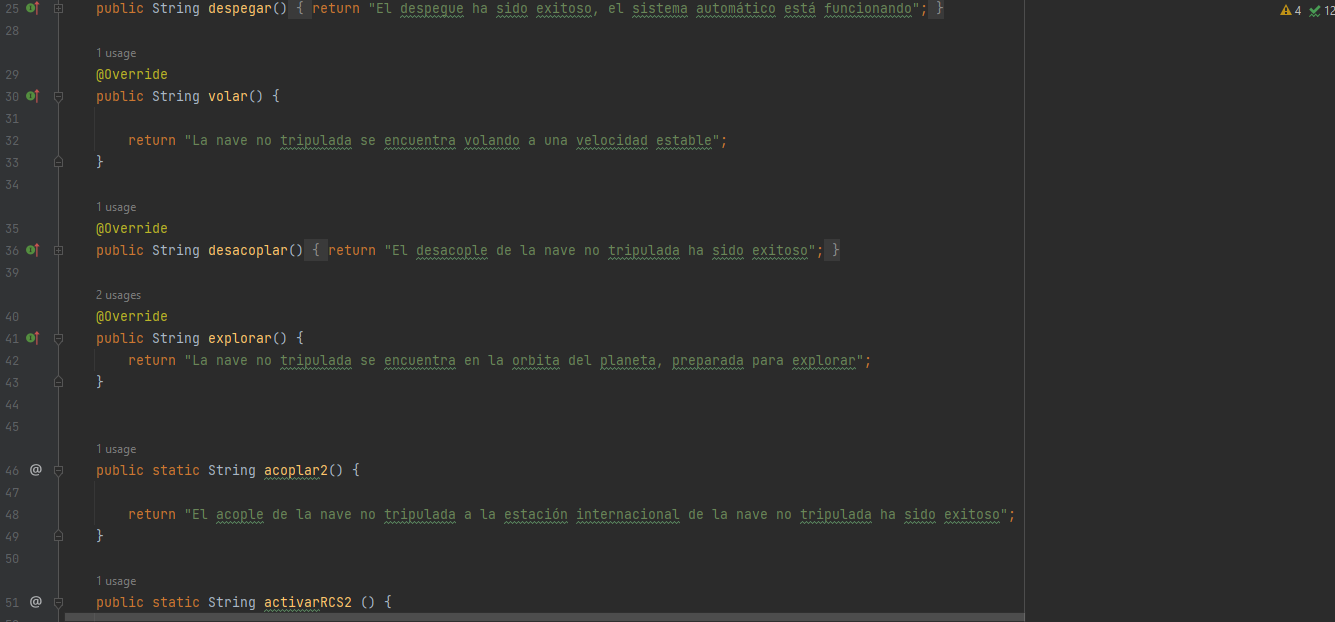
**Lanzadera**

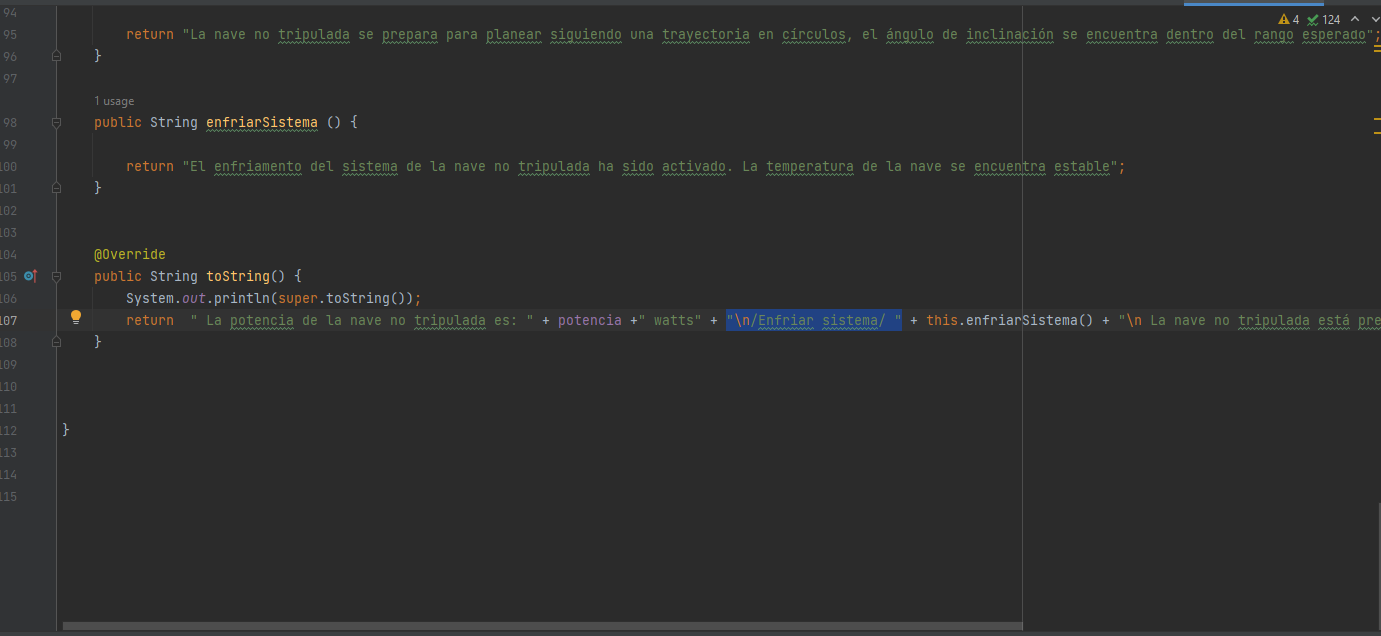




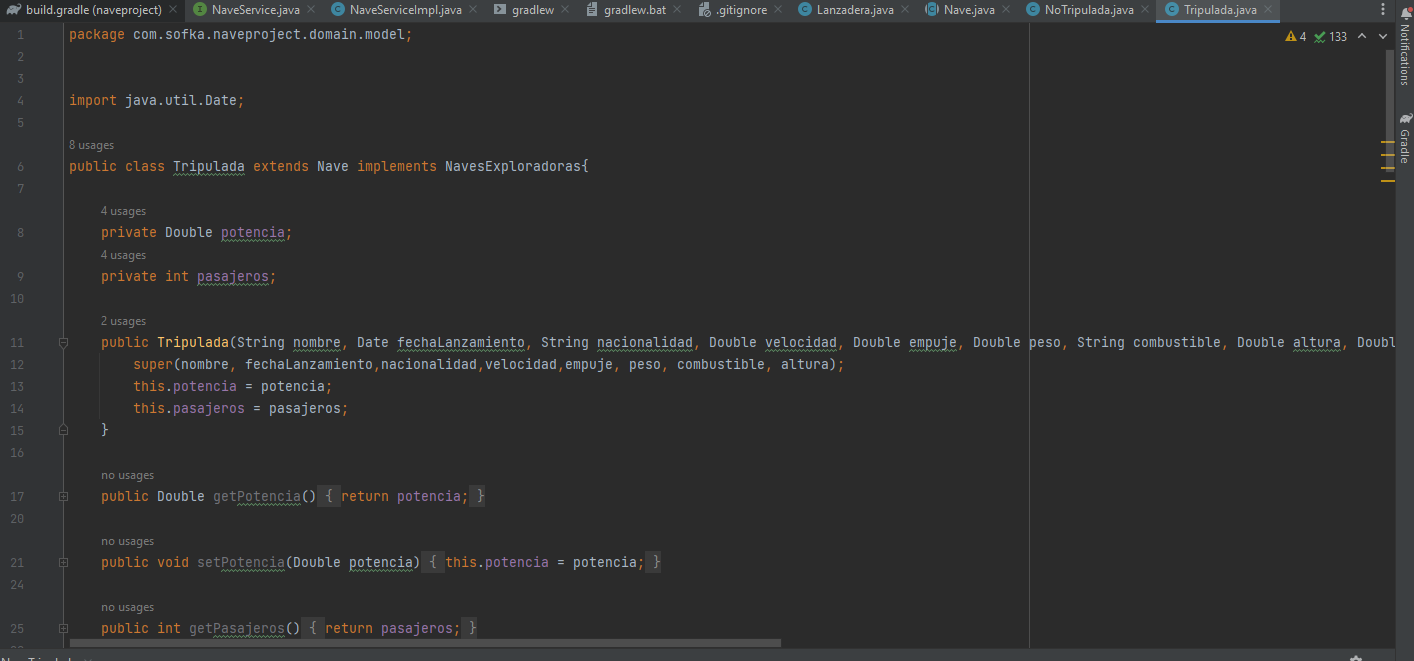
**NoTripulada**

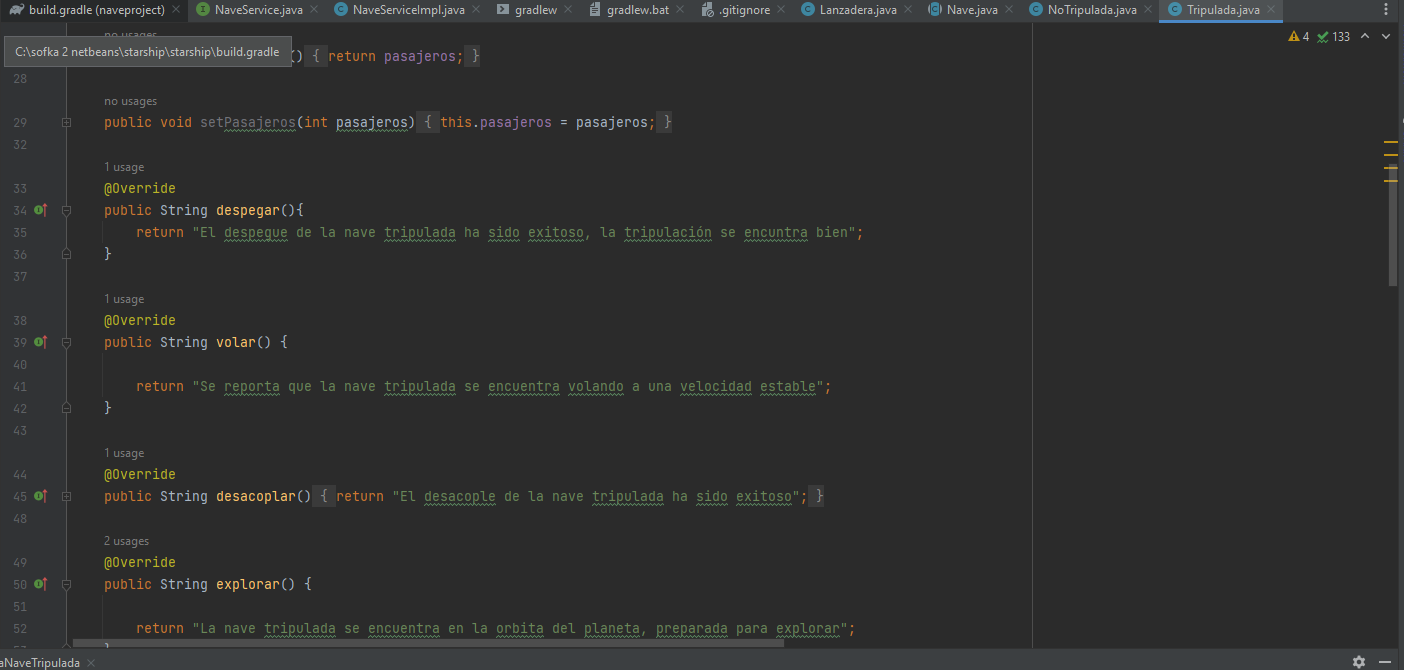


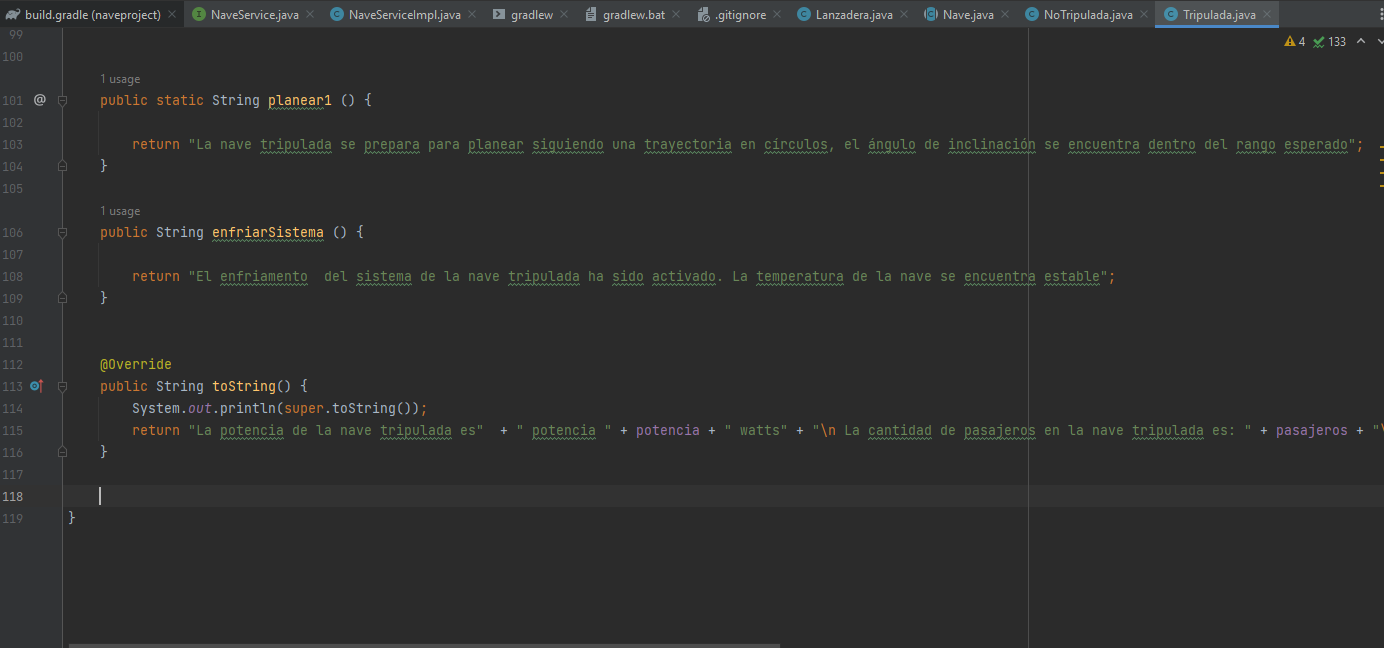


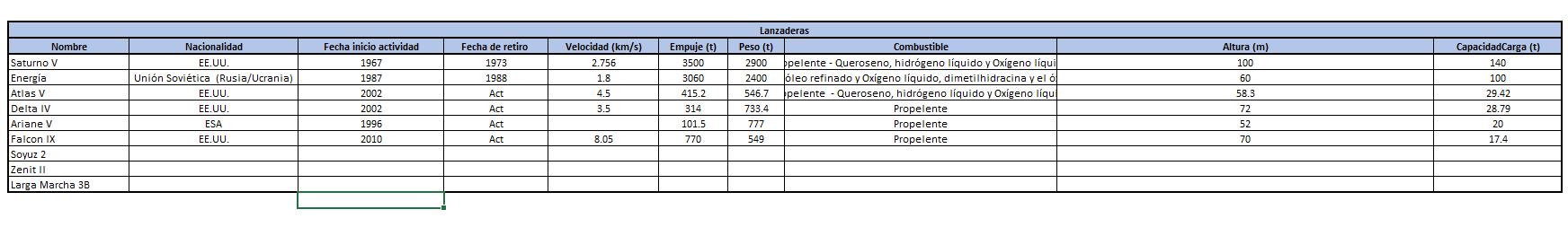


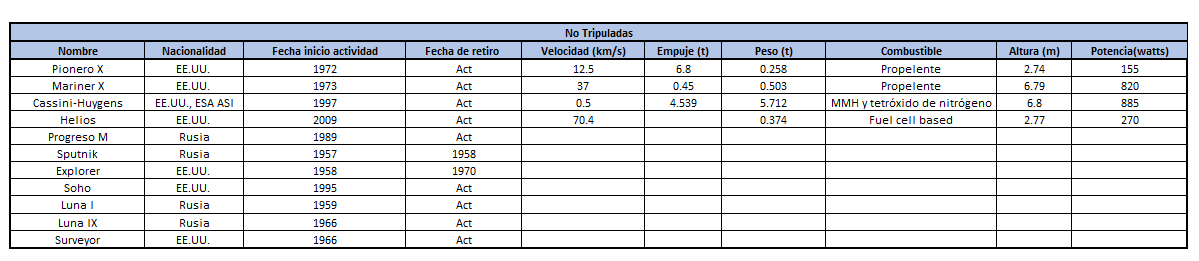
**Tripulada**

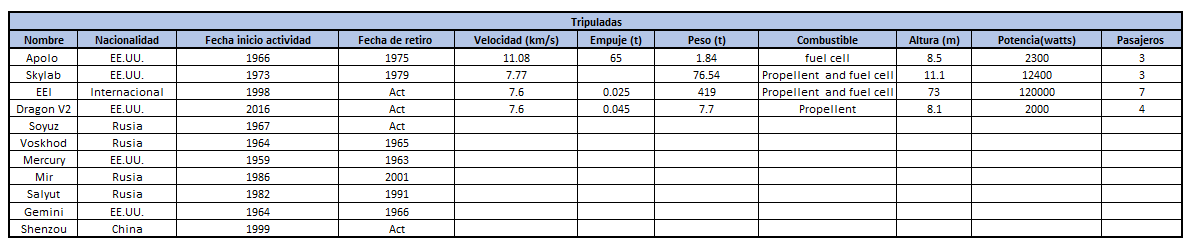




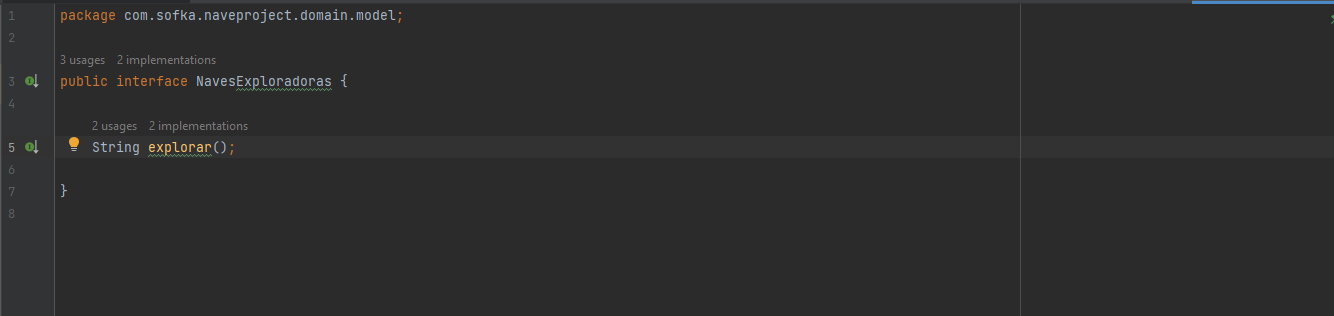




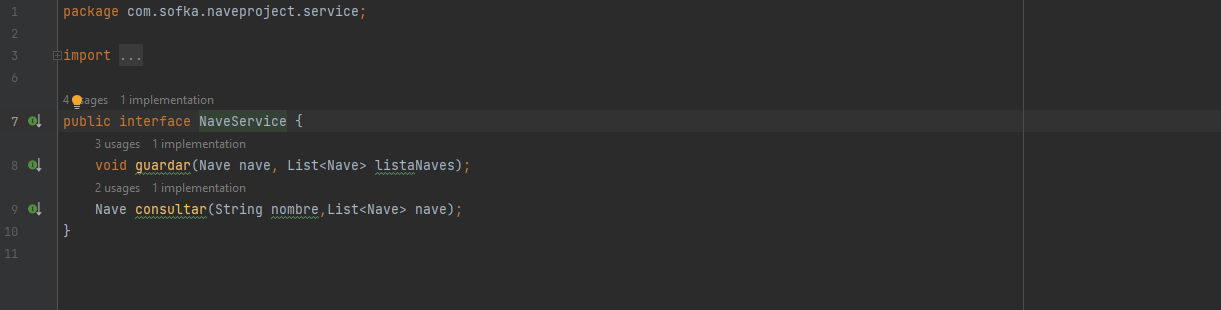




Por último se creó una interfaz titulada NavesExploradoras, ya que solo sería implementada por dos de las tres clases definidas, las tripuladas y las no tripulada, de esta forma se definió el método explorar

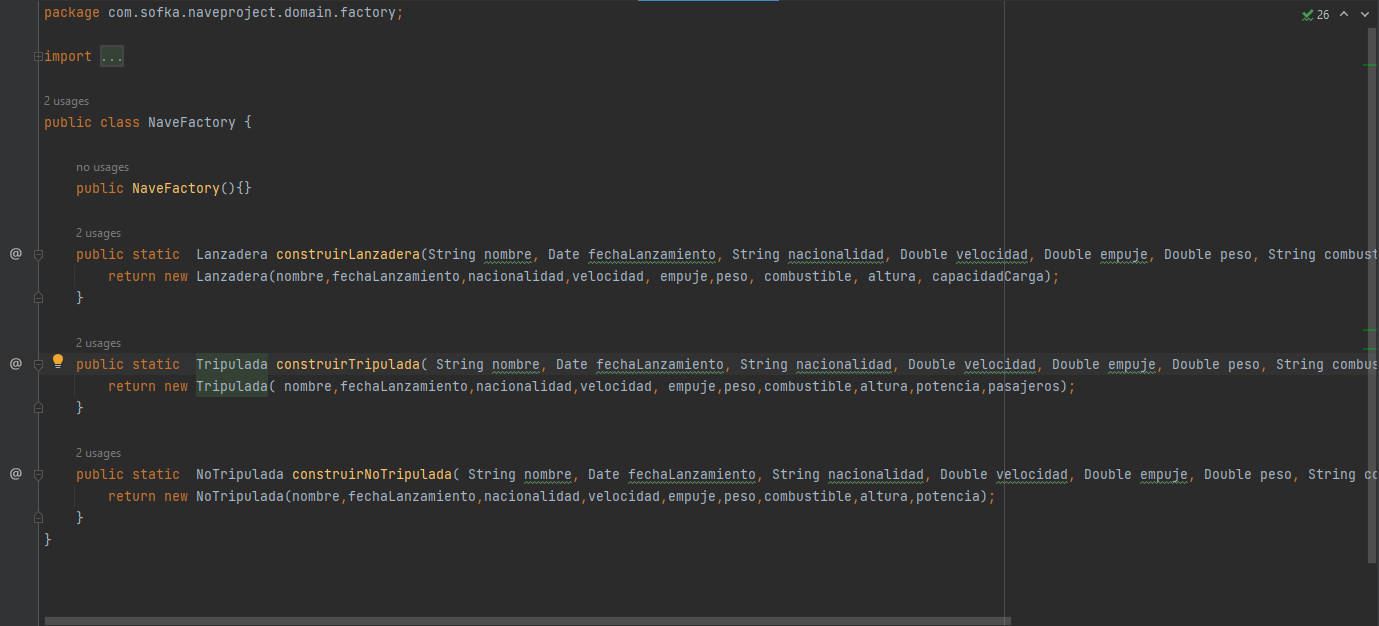


En la sección de controlador se definió el service, creando la interfaz NaveService e implementándola para abordar parte del CRUD, la creación y la consulta, para lo cual se utilizaron listas de tal forma que se vaya guardando la información en la lista



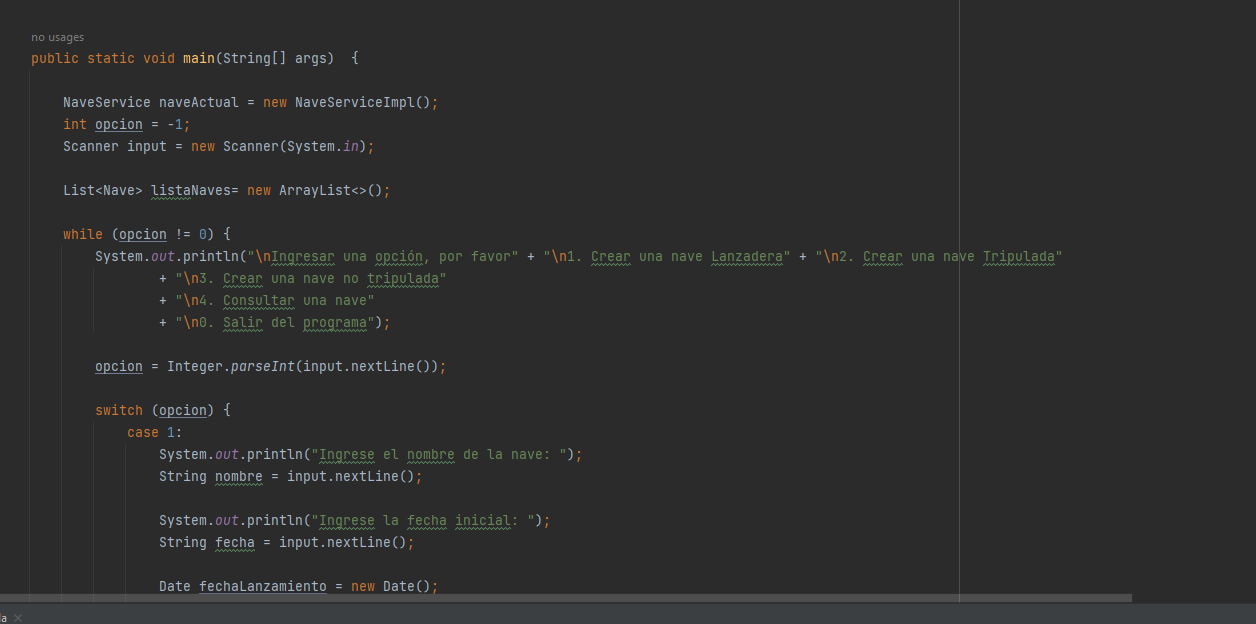


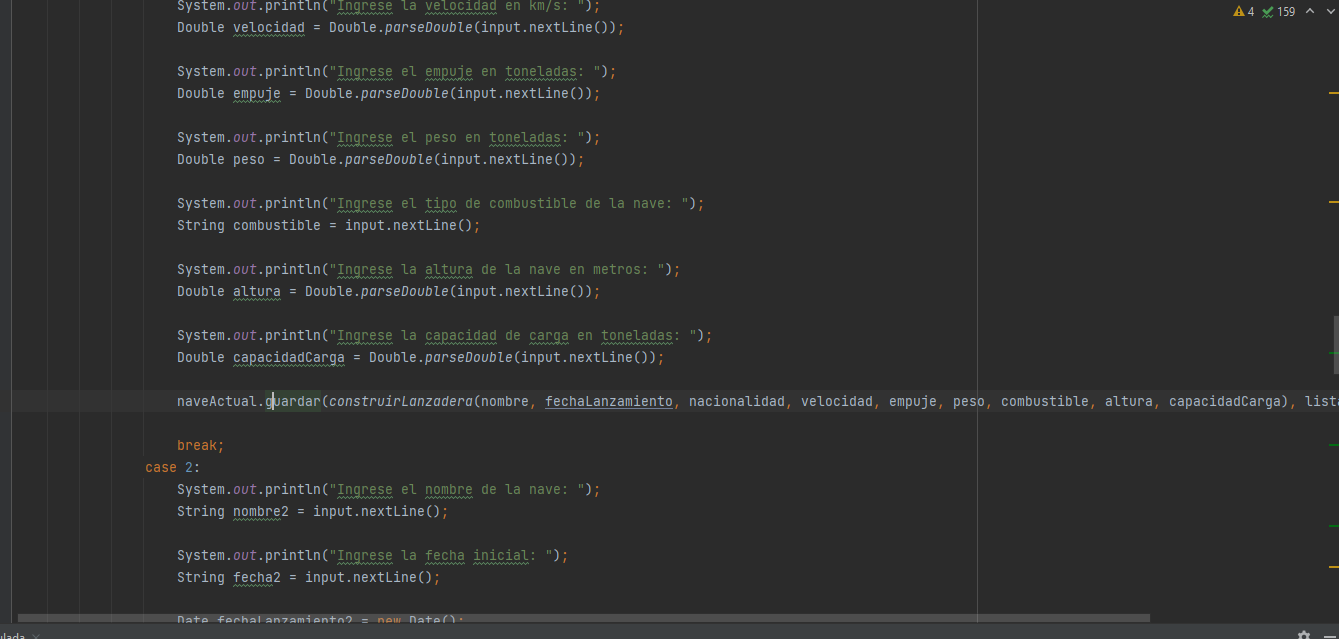
También se emplearon patrones de diseño Factory que lo que hace es crear métodos que de acuerdo a los parámetros que reciba construya un tipo particular de objeto, en este caso hay métodos para construir naves tipo lanzaderas, naves tripuladas y no tripuladas, son métodos estáticos es decir no requieren de la creación de un objeto para poder utilizarlas.

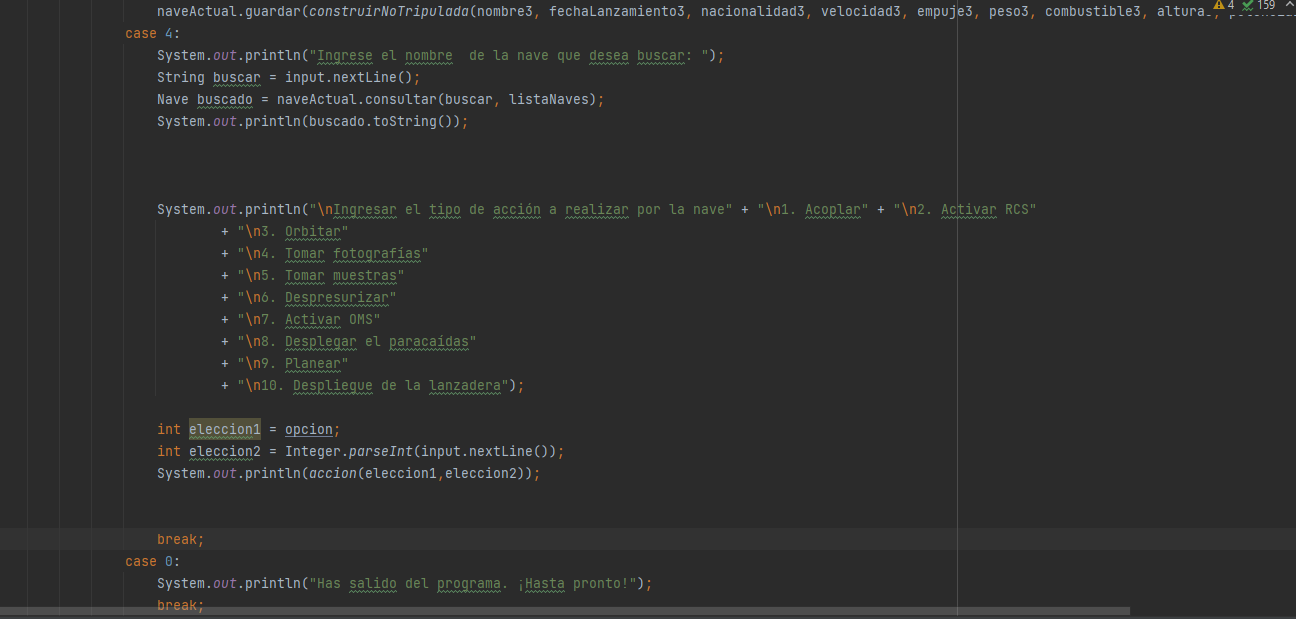


Por último, se abordó la interfaz de usuario, la cual se definió por consola, solicitando los datos al usuario a través de la consola registrándolos en listas y por medio de los patrones de diseño empleados anteriormente los puedo usar para clasificar según sea el tipo de nave y posteriormente utilizarlos con los métodos de guardar y consultar definidos en la carpeta de servicio. También se empleó el condicional de selección switch para definir varios casos.

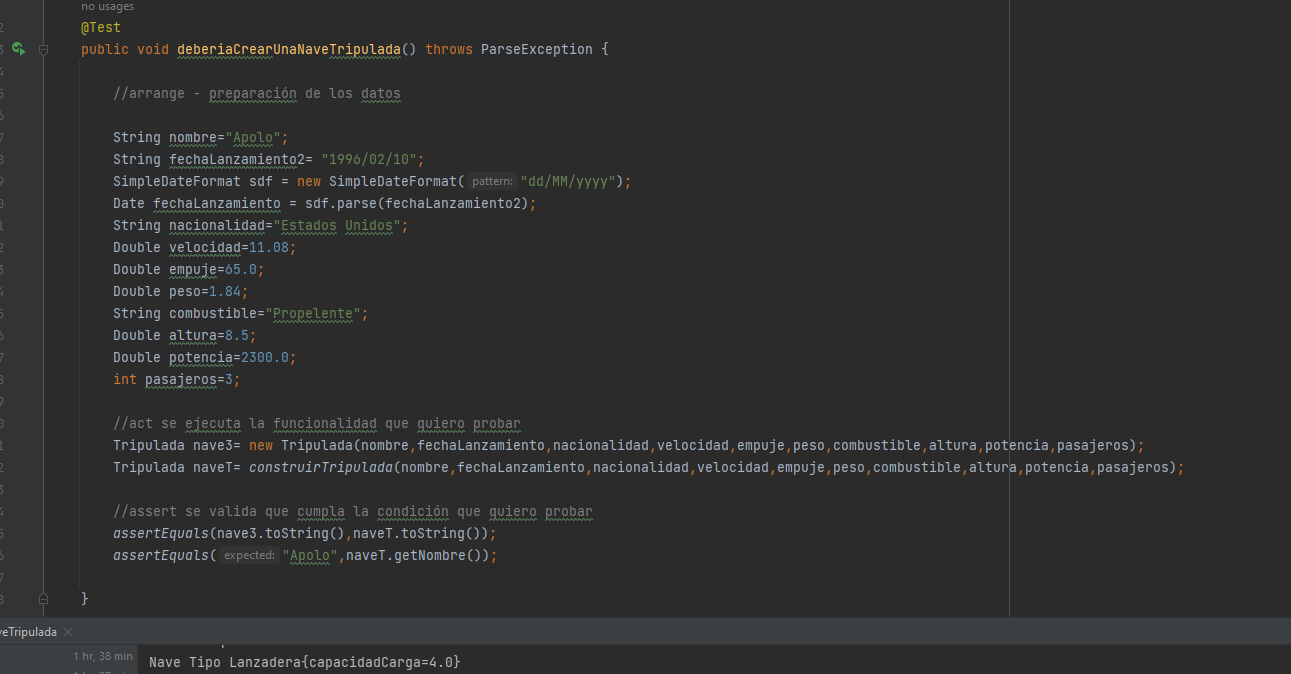
Guardar

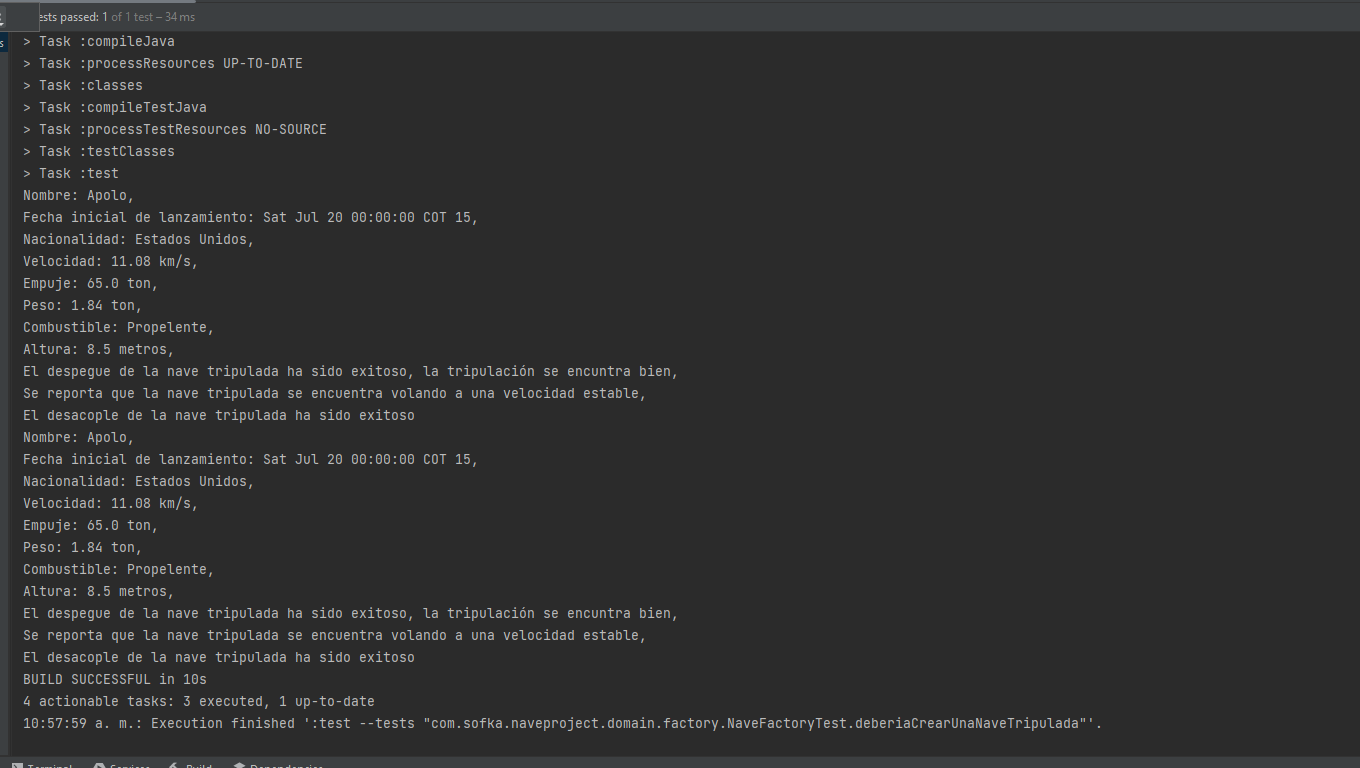






Por último, se realizaron las pruebas unitarias gracias a las dependencias descargadas de Spring Boot donde se evaluó que efectivamente se crearan cada una de las naves y se comparó con el nombre creado como se muestra a continuación





La ejecución final del programa se muestra a continuación:

