

MODULARIZACIÓN DE MAZMORRA

Introducción	3
Estructura del Proyecto	3
Construcción del JAR	3
Gestión de Dependencias	5
Conclusión	7

Introducción

Este documento detalla el proceso de modularización de la aplicación de mazmorras, enfocándose en la creación de archivos JAR para diferentes componentes del sistema. Se describen las decisiones de diseño, la estructura de los módulos y las interfaces de cada componente.

Estructura del Proyecto

El proyecto se ha dividido en varios módulos, cada uno encapsulando funcionalidades específicas. Los módulos principales orientados a interfaz son:

mTree.jar: Contiene las clases y métodos relacionados con la estructura de árbol de la mazmora.

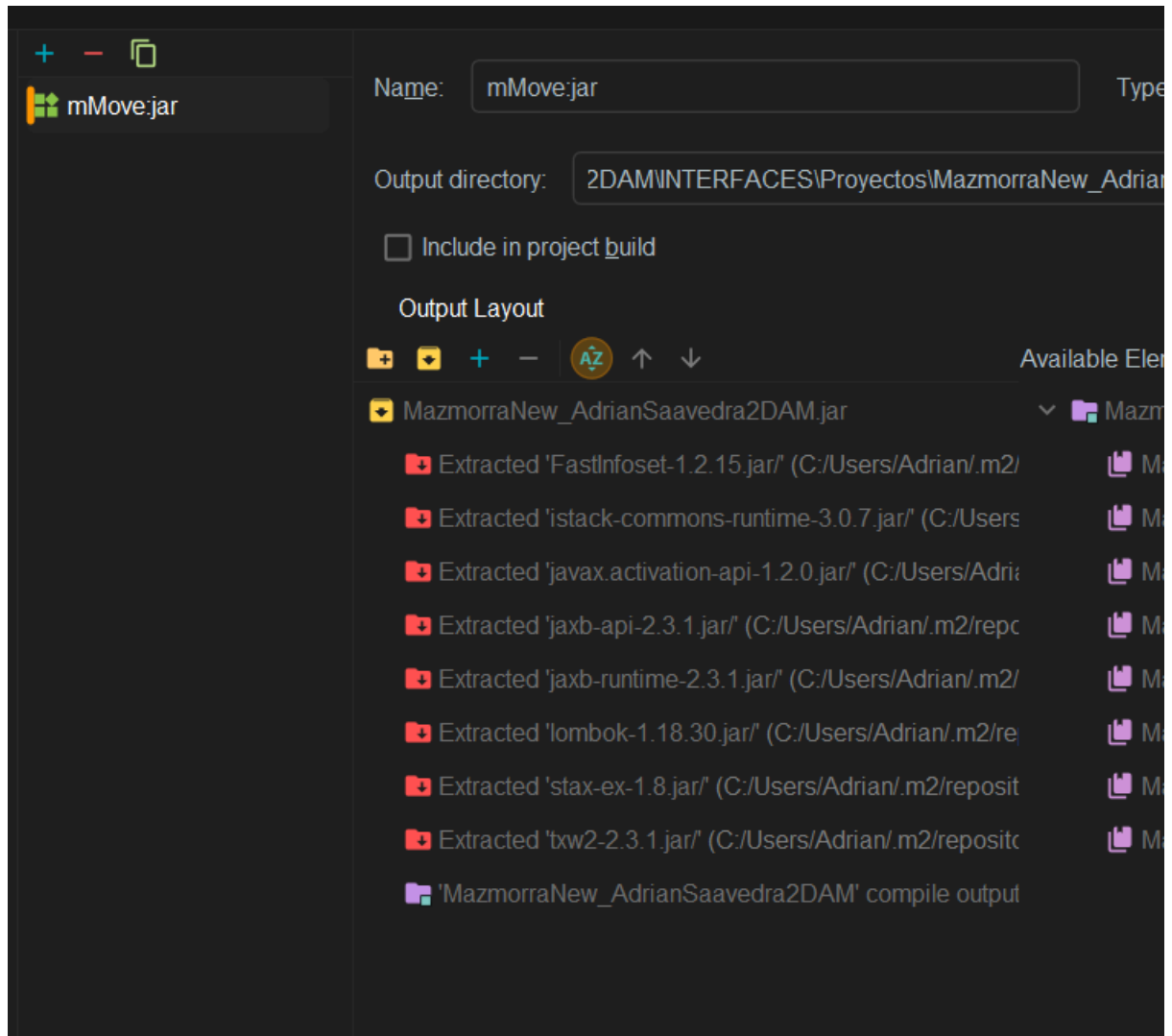
mMove.jar: Maneja la lógica de movimiento dentro de la mazmora.

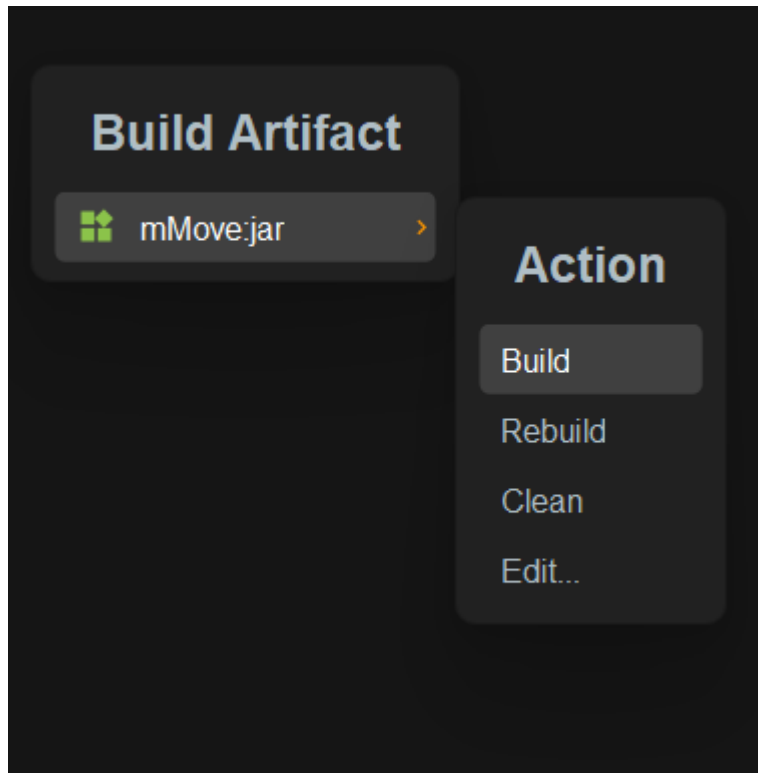
mLoad.jar: Se encarga de la carga de datos y recursos necesarios para la mazmora.

mLog.jar: Registra eventos y errores durante la ejecución.

Construcción del JAR

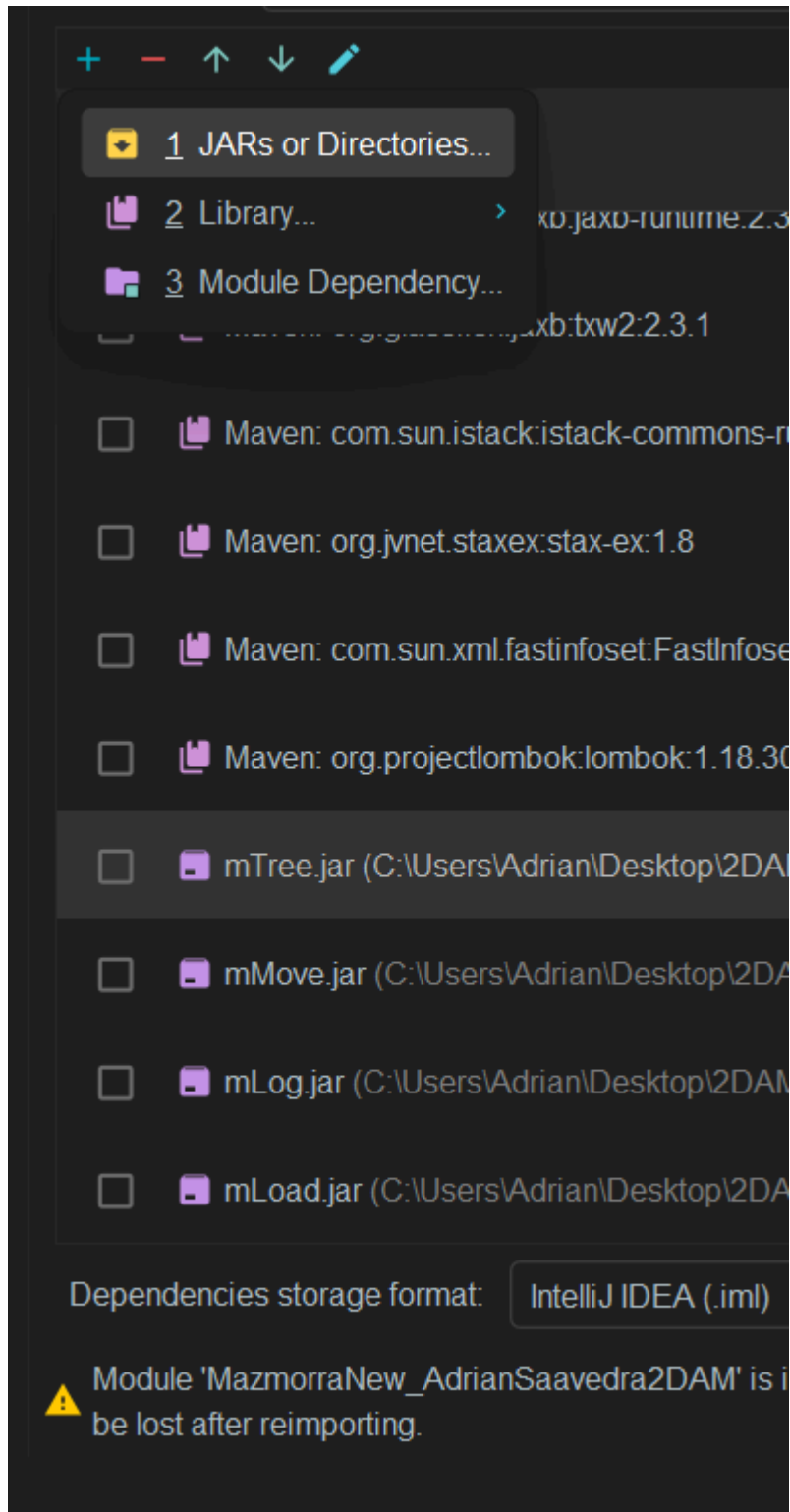
Se crea desde el Project Structures -> Artifacts en +, y luego utilicé la opción "Build" para compilar el artefacto mMove.jar, asegurando que todas las clases y recursos se puedan empaquetar correctamente.



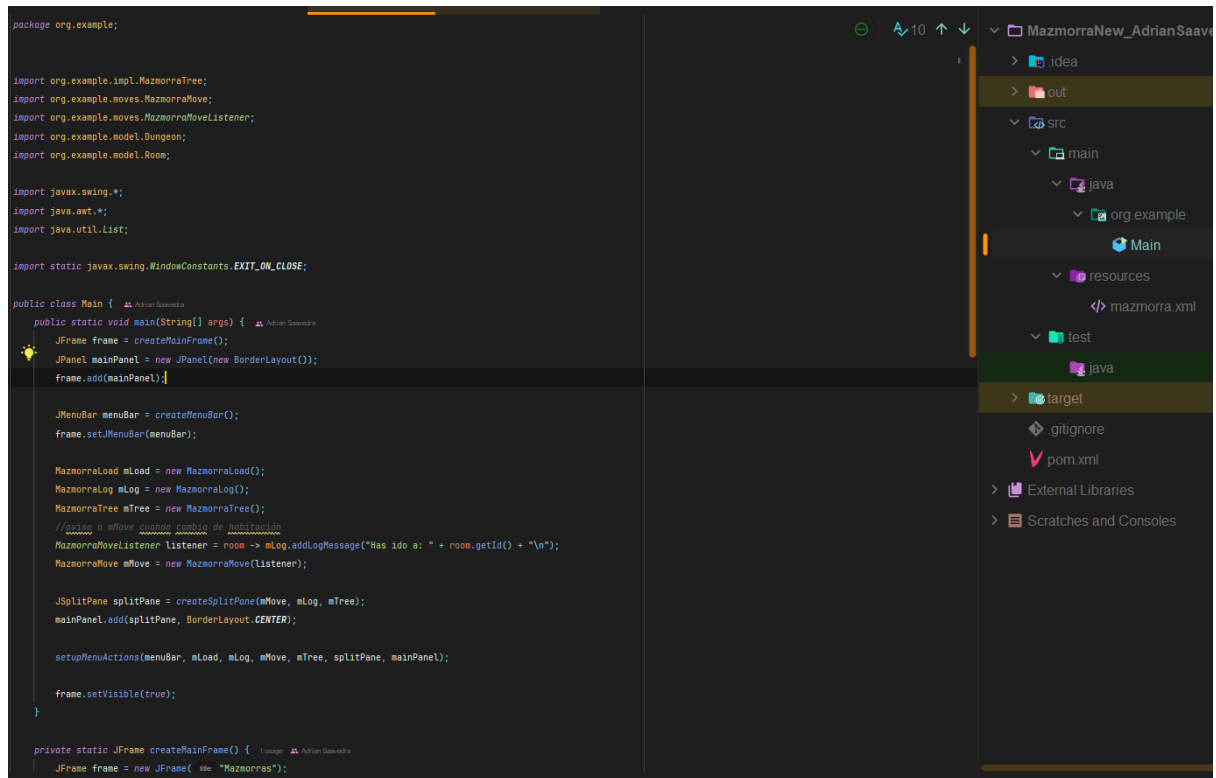


Gestión de Dependencias

En la configuración de dependencias del módulo, se añadieron los JARs necesarios, asegurando que cada módulo tuviera acceso a las bibliotecas requeridas.



Finalmente quedó así:



```
package org.example;

import org.example.impl.MazmorraTree;
import org.example.moves.MazmorraMove;
import org.example.moves.MazmorraMoveListener;
import org.example.model.Dungeon;
import org.example.model.Room;

import javax.swing.*;
import java.awt.*;
import java.util.List;

import static javax.swing.WindowConstants.EXIT_ON_CLOSE;

public class Main {
    public static void main(String[] args) {
        JFrame frame = createMainFrame();
        JPanel mainPanel = new JPanel(new BorderLayout());
        frame.add(mainPanel);

        JMenuBar menuBar = createMenuBar();
        frame.setJMenuBar(menuBar);

        MazmorraLoad mLoad = new MazmorraLoad();
        MazmorraLog mLog = new MazmorraLog();
        MazmorraTree mTree = new MazmorraTree();
        //aviso a mMove cuando cambia de habitación
        MazmorraMoveListener listener = room -> mLog.addLogMessage("Has ido a: " + room.getId() + "\n");
        MazmorraMove mMove = new MazmorraMove(listener);

        JSplitPane splitPane = createSplitPane(mMove, mLog, mTree);
        mainPanel.add(splitPane, BorderLayout.CENTER);

        setMenuActions(menuBar, mLoad, mLog, mMove, mTree, splitPane, mainPanel);

        frame.setVisible(true);
    }

    private static JFrame createMainFrame() {
        JFrame frame = new JFrame("Mazmorras");
    }
}
```

The screenshot shows an IDE with a Java file named Main.java. The code is a Java Swing application for a dungeon game. It imports various classes from the org.example package, including MazmorraTree, MazmorraMove, MazmorraMoveListener, Dungeon, Room, and MazmorraLoad. The main method creates a JFrame, a JPanel, a JMenuBar, and a JSplitPane. It then initializes several objects: MazmorraLoad, MazmorraLog, MazmorraTree, MazmorraMoveListener, and MazmorraMove. The code also includes comments in Spanish, such as "aviso a mMove cuando cambia de habitación". The project explorer on the right shows the project structure, including the org.example package, Main class, resources, mazmorra.xml, test, java, target, gitignore, pom.xml, External Libraries, and Scratches and Consoles.

Conclusión

La modularización de la aplicación de mazmorras ha mejorado la organización y mantenibilidad del código. Al dividir el proyecto en módulos específicos, se facilita el desarrollo y la integración de nuevas funcionalidades, optimizando la gestión de dependencias y permitiendo una mayor flexibilidad para futuras expansiones. En resumen, esta estrategia es esencial para construir aplicaciones robustas y escalables.