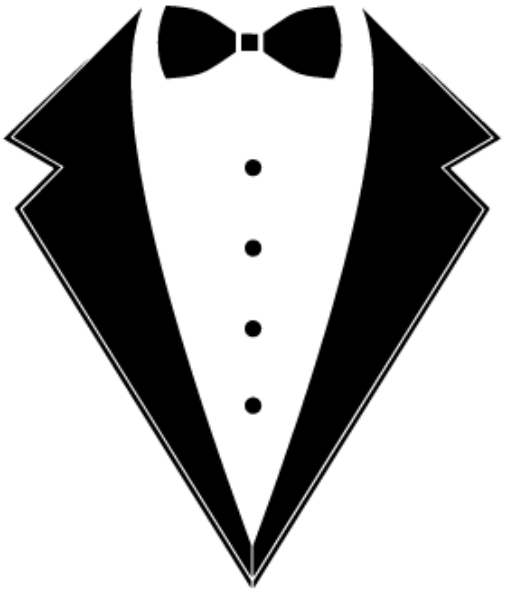
Manual Técnico



DONCOMANDA

ÍNDICE

1. Introducción

En este manual se proporcionarán detalles para el mantenimiento, implementación e instalación de la aplicación DonComanda. Se recomienda acceder al repositorio de GitHub donde se encontrarán todos los archivos necesarios, tanto de código fuente como de archivos .sql, .fxml, JavaDoc…

2. Creación del software

Para la creación de de DonComanda, se ha utilizado el lenguaje de programación ***Java*** el cuál se ha conectado a una base de datos local creada con ***MySQL***. Para la gestión desde java de la base de datos se ha utilizado ***JDBC***. Mientras que para la parte visual se ha utilizado la librería de ***JavaFX*** con la ayuda del entorno de programación ***IntelliJ IDEA***, que ofrece una interfaz gráfica para crear las vistas.

También se ha utilizado la librería de ***JasperReport*** en Java para poder generar pdfs, mientras que para el diseño de los archivos .jrxml se ha utilizado el entorno de desarrollo de ***JasperSoft***, que ofrece la posibilidad de crear datasets con JDBC para generar tablas con datos obtenidos de la base de datos a través de consultas.

3. Diagramas

3.1 Diagrama de Entidad Relación

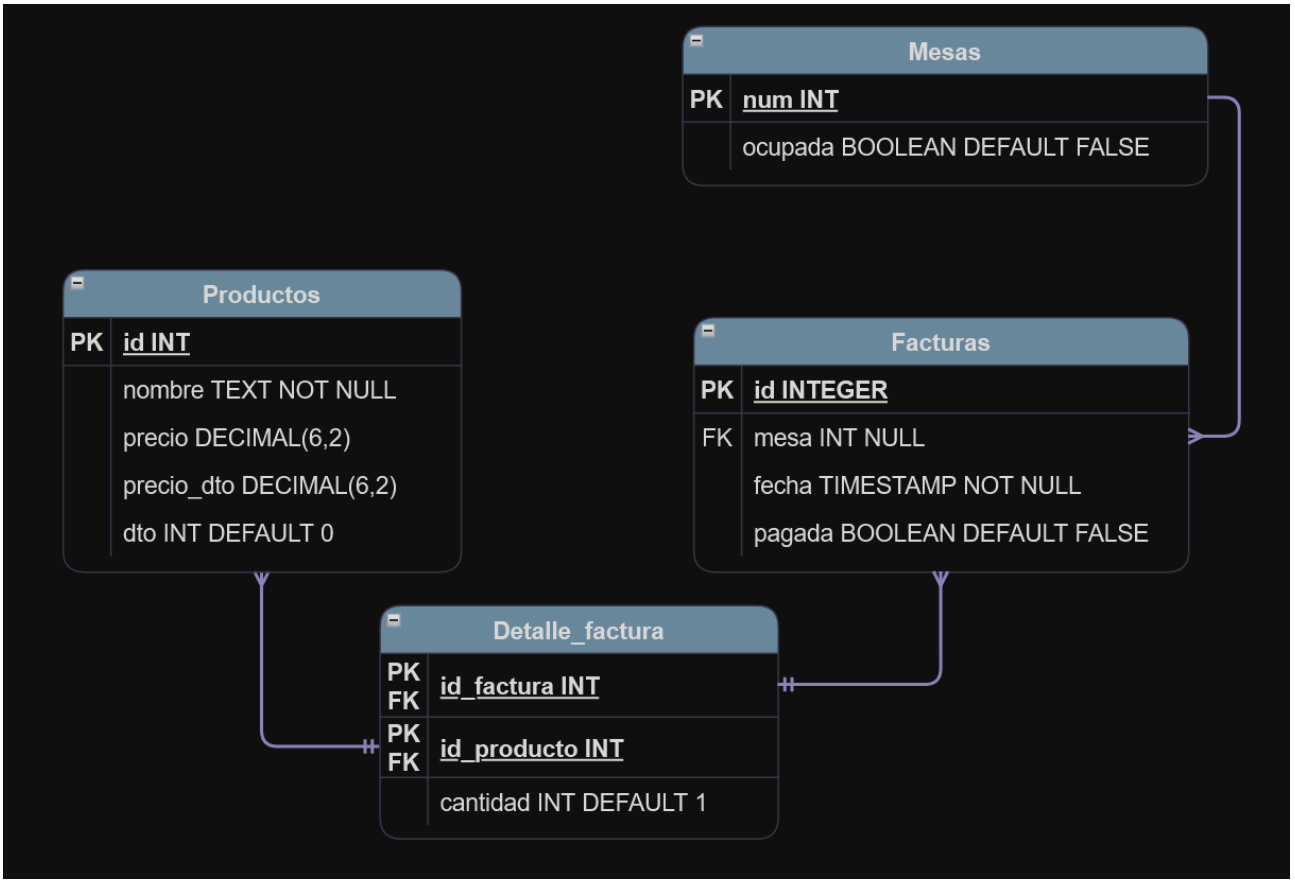


Figura 1: Diagrama de Entidad Relación creado en Draw.io

En el diagrama se presentan las diferentes entidades, sus atributos y las relaciones entre ellos, que se han creado para generar la base de datos en MySQL. De forma más detallada, las entidades tienen las siguientes funciones:

- **MESAS** → Almacena el número de mesa que le corresponde a la mesa, así como si está ocupada o no. Mantiene una relación 1:N con la entidad **facturas**.
- **FACTURAS** → Almacena datos relacionados con la factura, como es la fecha de creación, si está pagada y si tiene una mesa relacionada o no (ya que puede tomar valor *null*). Mantiene dos relaciones, una N:1 con la entidad **mesas** y una N:M con la entidad **productos**.
- **PRODUCTOS** → Almacena los datos básicos de un producto, algo a destacar sería su atributo *precio_dto* que es básicamente un atributo computado, que lo que hace es calcular su precio en base al atribtuo *precio* y *dto*. Mantiene una relación N:M con la entidad **facturas**, generando una nueva tabla **detalle_factura**.
- **DETALLE_FACTURA** → En pocas palabras, indica qué cantidad de productos se ha pedido y a qué factura pertenece ese pedido.

El código para generar dicha base de datos y tablas es el siguiente:

```
CREATE DATABASE IF NOT EXISTS DonComanda;
USE DonComanda;

CREATE TABLE mesas(
   num INT PRIMARY KEY,
   ocupada BOOLEAN DEFAULT FALSE
);

CREATE TABLE productos(
   id INT PRIMARY KEY AUTO_INCREMENT,
   nombre TEXT NOT NULL,
   precio DOUBLE,
   precio_dto DECIMAL(6,2) AS (precio-(precio*(dto/100))),
   dto INT DEFAULT 0
);

CREATE TABLE facturas(
   id INT PRIMARY KEY AUTO_INCREMENT,
   mesa INT NULL,
   fecha TIMESTAMP NOT NULL,
   pagada BOOLEAN DEFAULT FALSE,
   FOREIGN KEY (mesa) REFERENCES mesas(num)
);

CREATE TABLE detalle_factura(
   id_factura INT,
   id_producto INT,
   cantidad INT DEFAULT 1,
   PRIMARY KEY (id_factura, id_producto),
   FOREIGN KEY (id_factura) REFERENCES facturas(id) ON DELETE CASCADE,
   FOREIGN KEY (id_producto) REFERENCES productos(id) ON DELETE CASCADE
);
```

Código 1: Código respectivo para crear la base de datos en MySQL

3.2 Diagrama de casos de uso

En este diagrama se detalla el papel a desempeñar en relación con la aplicación por parte de las personas relacionadas, en este caso sólo hay un actor: el camarero.

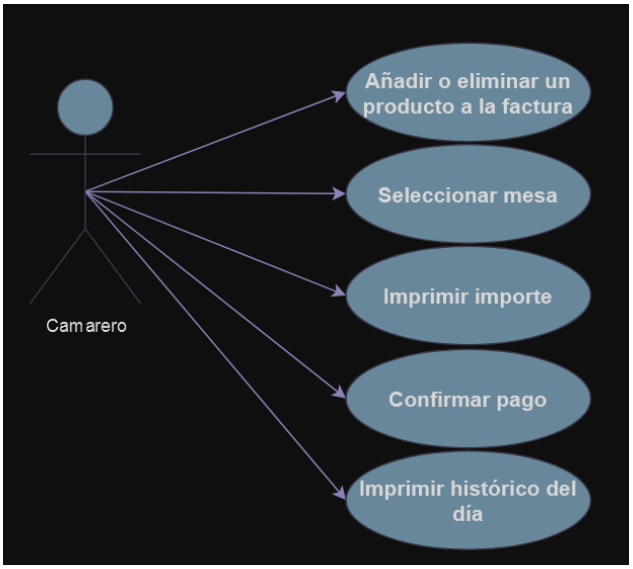


Figura 2: Diagrama de casos de uso

3.3 Diagrama de flujo

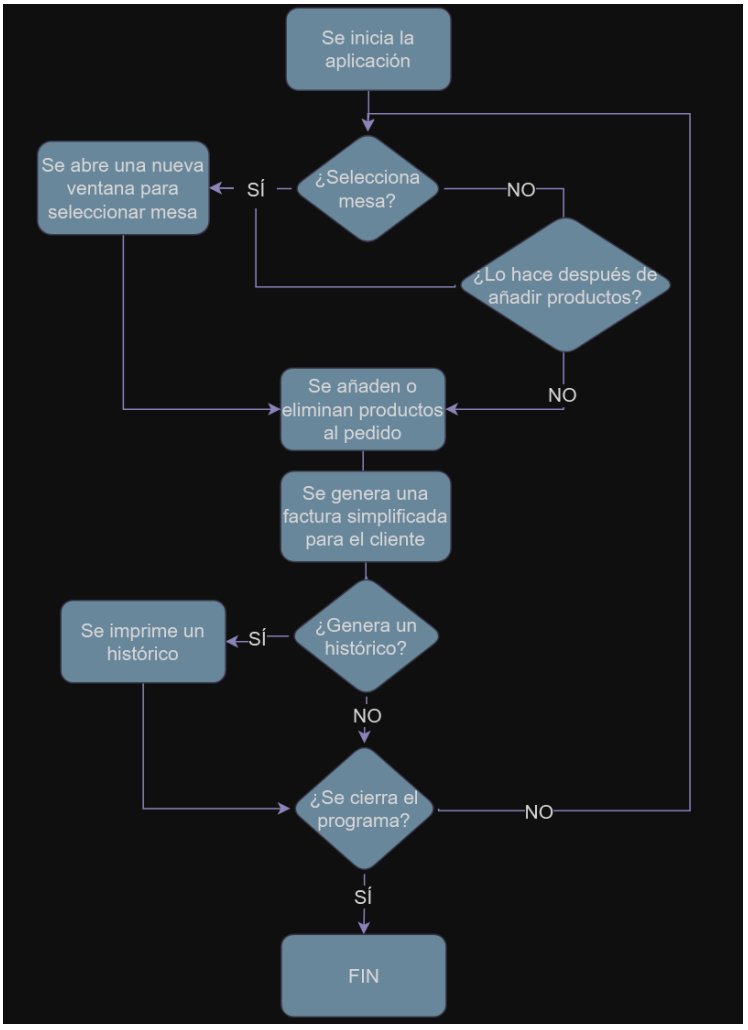


Figura 3: Diagrama de flujo

4. Creación del ejecutable

El ejecutable no ha sido posible de crear con la funcionalidad de JasperReport, ya que uno de los pasos para poder crear el ejecutable es eliminar un archivo que conecta la librería con el proyecto. Sin embargo, los pasos a seguir para poder crear el ejecutable son:

4.1 Cambiar versión del compilador

Tendremos que ir al archivo ***pom.xml*** del proyecto para cambiar la versión del compilador de maven de la que tengamos a la **1.8**, pues es la versión necesaria para generar el archivo .jar funcional:

```
<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

También tendremos que cambiarlo en los plugins:

```
<build>
   <plugins>
       ...
       <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-compiler-plugin</artifactId>
           <version>3.8.0</version>
           <configuration>
               <release>1.8</release>
           </configuration>
       </plugin>
       ...
   </plugins>
</build>
```

Después de hacer esos cambios haremos click en el símbolo que aparecerá arriba a la derecha del editor para cargar los cambios realizados en Maven:

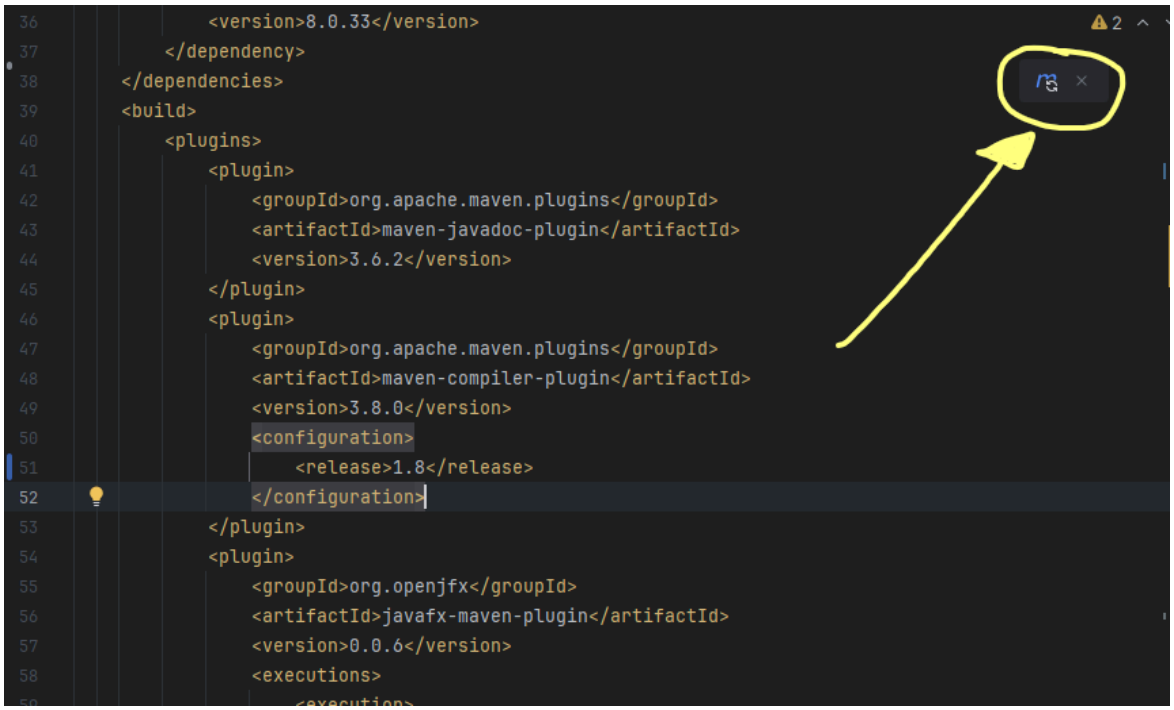


Figura 4: Cargar cambios en Maven

4.2 Eliminar archivo "module-info.java"

Después de realizar el paso anterior, el IDE nos indicará que tenemos un error en el archivo **module-info.java**, que se encuentra en la ruta "*src/main/java/module-info.java*", debido a que java 8 no soporta módulos. Por ello, tendremos que eliminar el archivo.

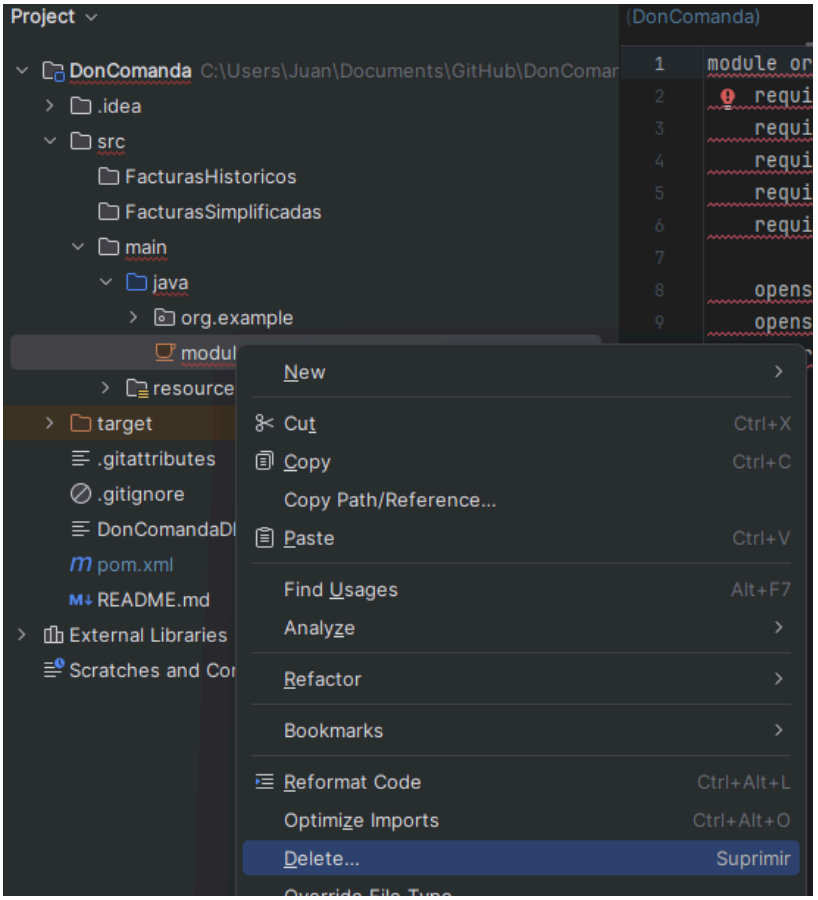


Figura 5: Eliminar module-info.java

4.3 Generar artefacto (crear .jar)

Ahora generamos los artefactos que se necesitan más adelante para poder crear el archivo .jar. Para ello, tendremos que seguir los siguientes pasos:

1. Abriremos la estructura del proyecto ya sea con el atajo *Ctrl+Alt+Shift+S* o bien desde la pestaña *File > Project Structure…*

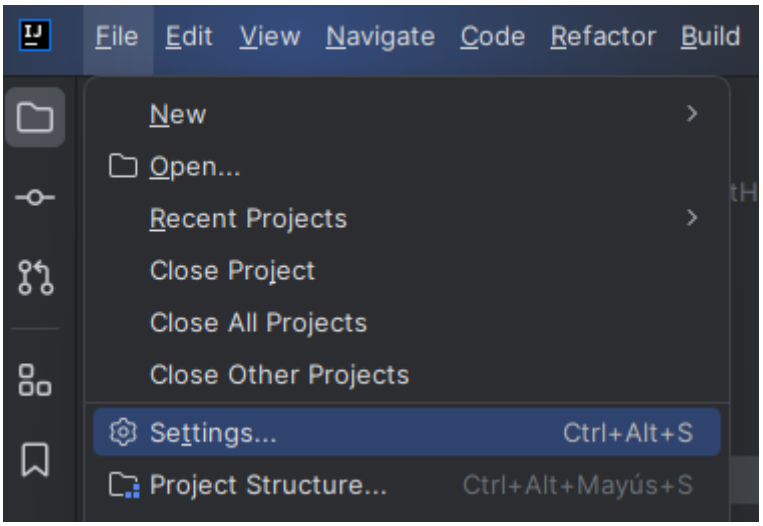


Figura 6: Abrir la estructura del proyecto

2. Se abrirá una nueva ventana, y en ella seleccionamos *Project Settings > Artifacts > + > JAR > From modules with dependencies…*

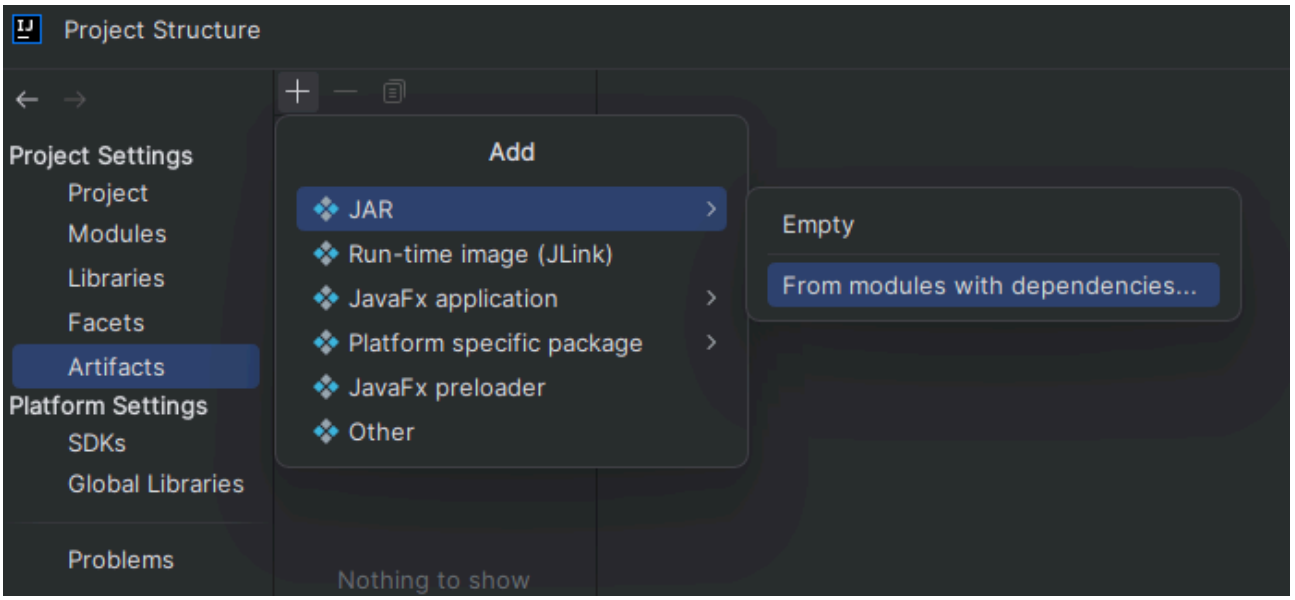


Figura 7: Añadir un Jar desde un módulo con dependencias

3. Se abrirá otra ventana más pequeña, aquí seleccionamos la clase que contiene el main que lanza el proyecto, para ello simplemente en la opción *Main Class:* seleccionamos la carpeta y la clase que tiene el main

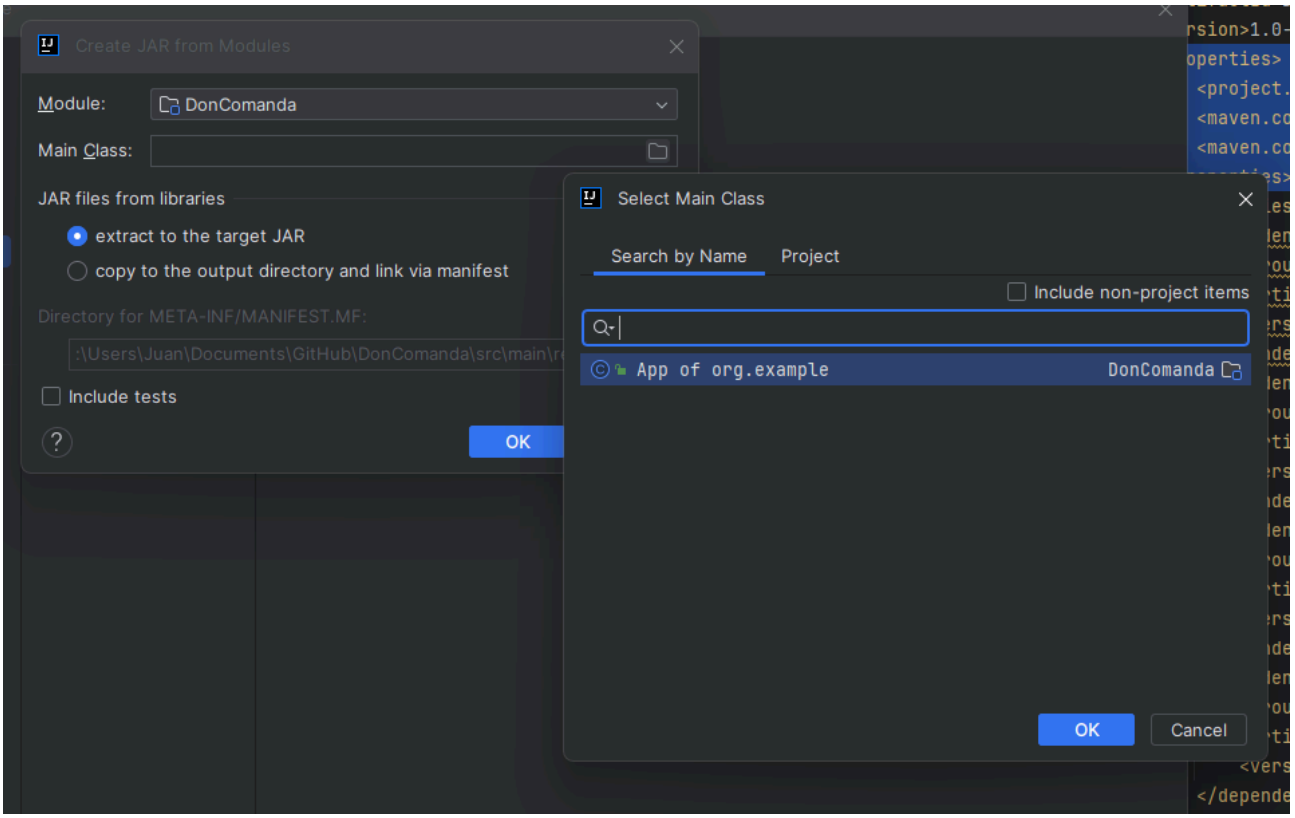


Figura 8: Seleccionar la clase con el Main

4. Tras darle a ok en las dos últimas ventanas, habrá generado el artefacto con las dependencias. Por tanto, sólo quedará darle a *Apply* y finalmente a *Ok*.

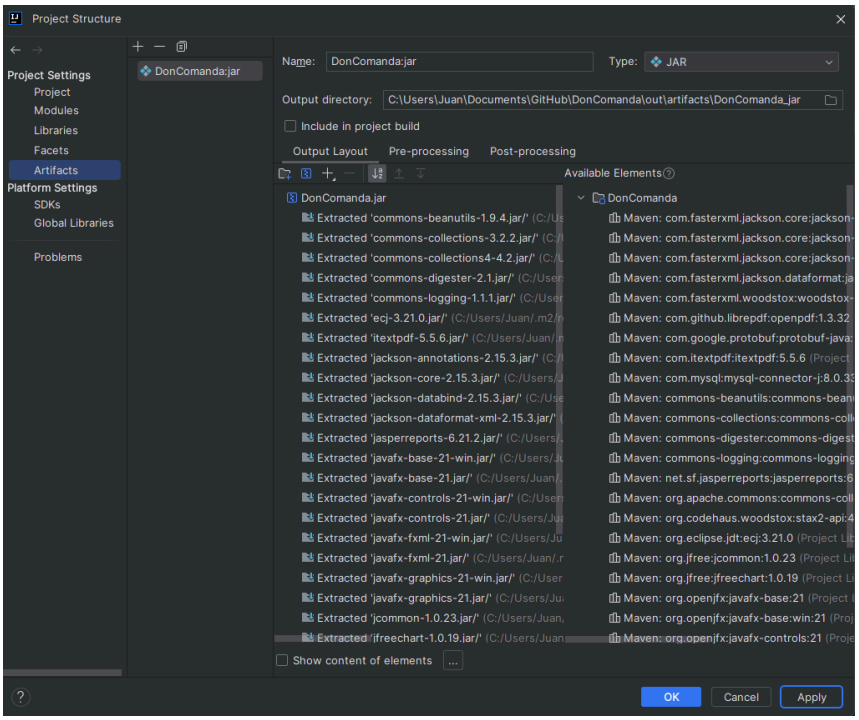


Figura 9: Aplicar la creación del artefacto

5. Finalmente, queda crear la build del artefacto, para ello volveremos a las pestañas y seleccionamos *Build* > *Build artifacts*

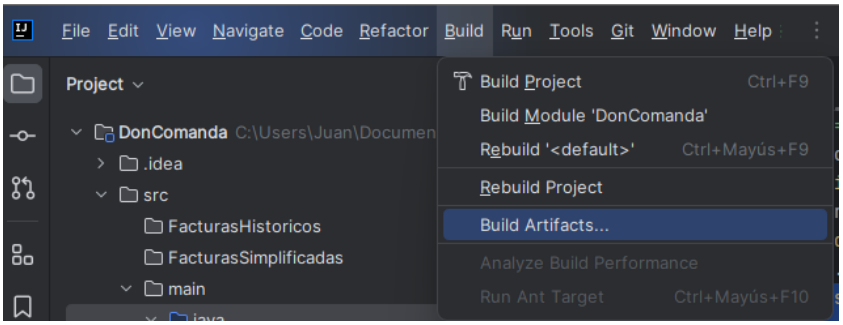


Figura 10: Crear el .jar

6. Encontramos el .jar en tras adentrarnos en unas carpetas generadas, la ruta desde la raíz del proyecto sería "*out/artifacts/DonComanda_jar/DonComanda.jar*"

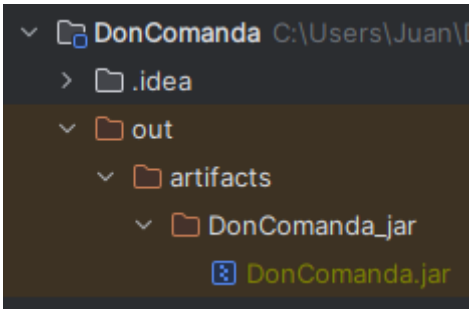


Figura 11: Ubicación del .jar

4.4 Generar el ejecutable

Para ello se necesitará la aplicación ***Launch4J*** y ***JRE 1.8*** (Java Runtime Environment). Con lo anterior mencionado instalado seguiremos los siguientes pasos:

1. Indicar el nombre del output y la ruta del .jar

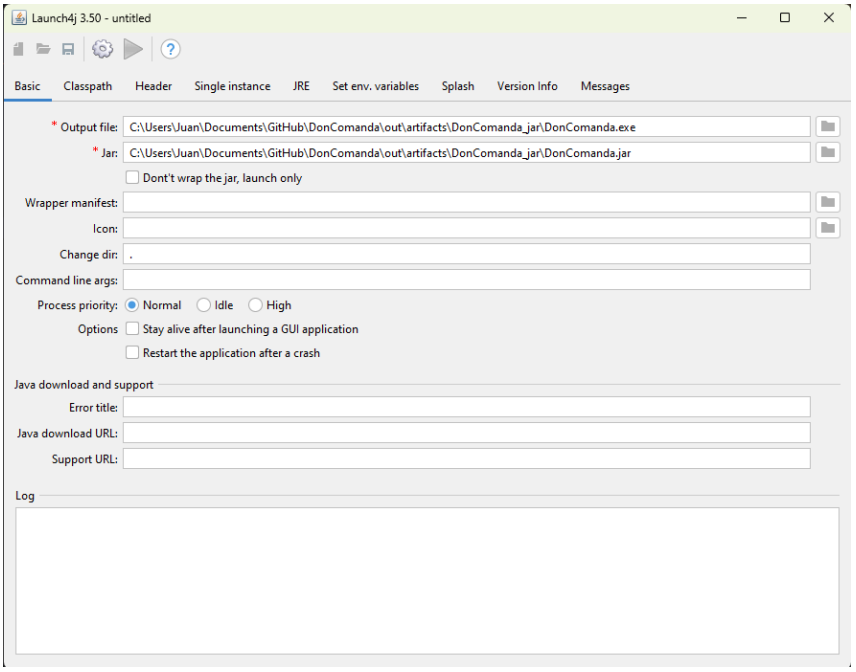


Figura 12: Indicar output del .exe y ruta del .jar

2. Cambiaremos a la pestaña *JRE* donde indicaremos la ruta de la carpeta /bin del jre, es decir, que tendremos una ruta parecida a "*C:\Program Files\Java\jre-1.8\bin*"

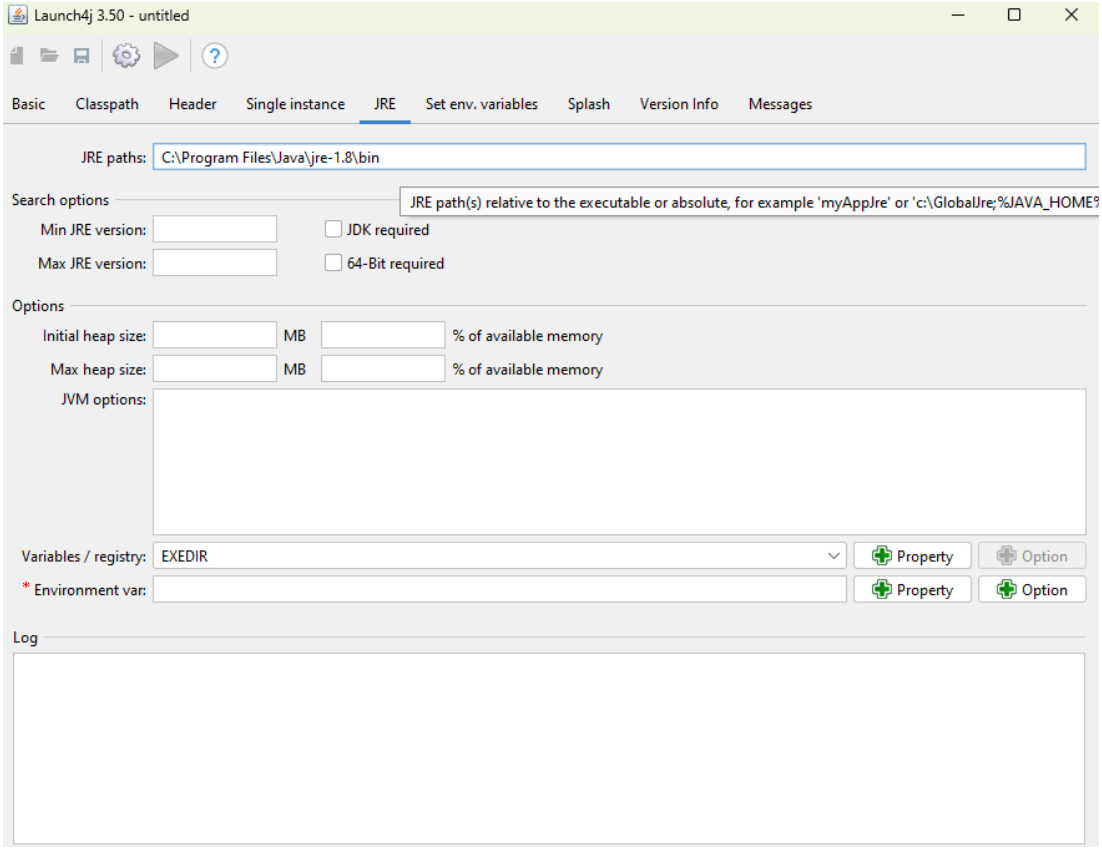


Figura 12: Indicar la ruta a la carpeta bin de jre

3. Clicamos sobre la imagen del engranaje y nos pedirá una ubicación y un nombre para la ruta del .xml que contiene la configuración para generar el ejecutable.

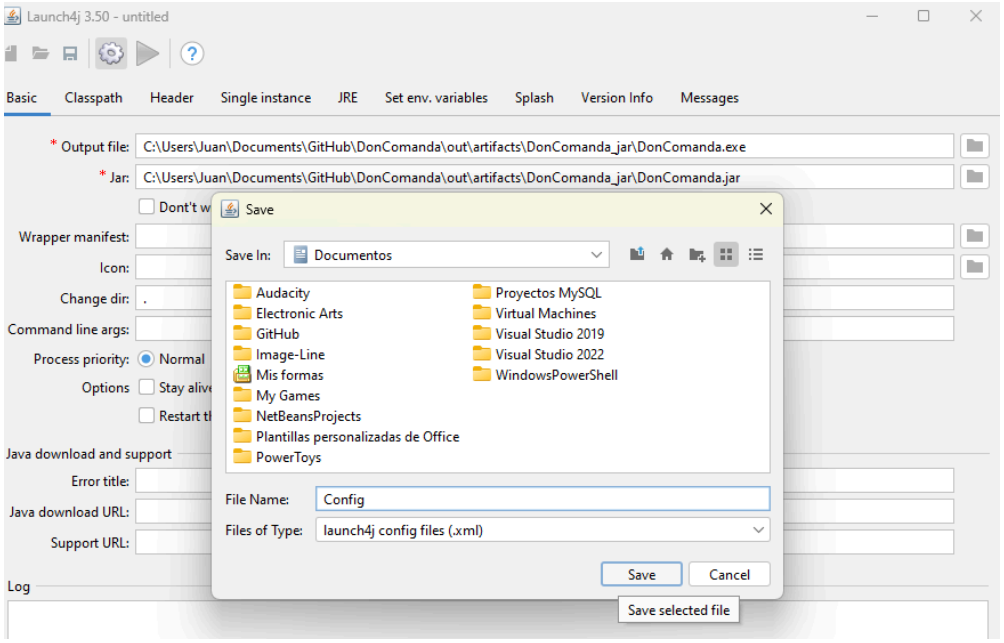


Figura 13: Guardar la configuración necesaria para crear el .exe

4. Finalmente hacemos clic sobre la imagen del play azul y ya tendremos el ejecutable en la ruta que habíamos indicado anteriormente.

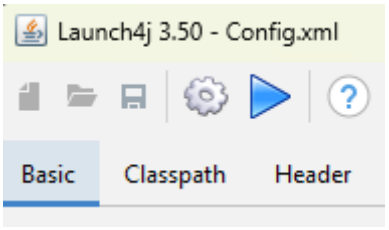


Figura 14: Generar .exe clicando sobre el play azul