

Programação Avançada

Inclusão do *JavaFX* com *Maven*

Maven

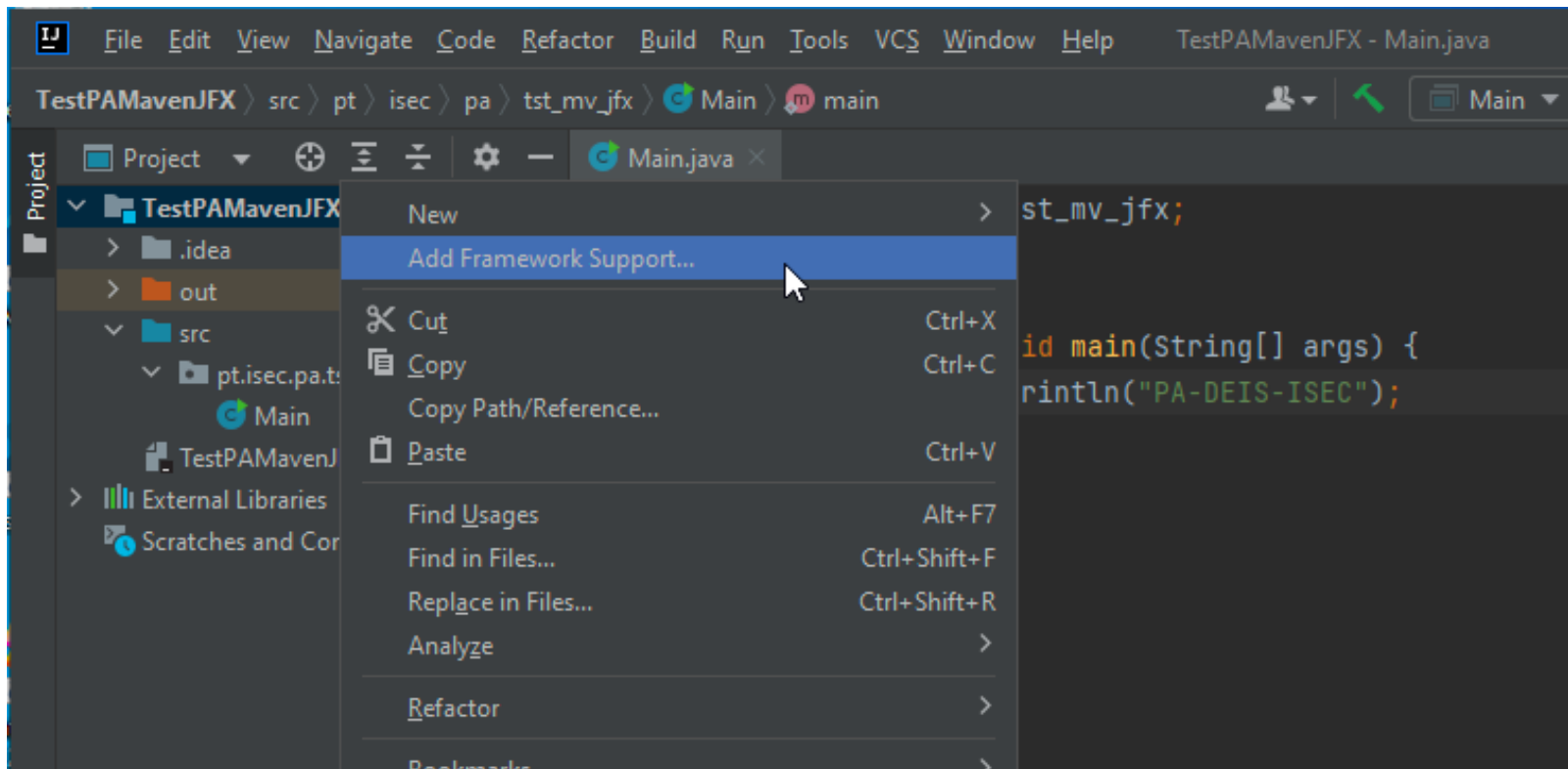
- O *Maven* é uma ferramenta de automatização de processos de compilação de projetos de software
- A sua principal utilização surge no âmbito de projetos *Java*, embora possa ser usado no contexto de vários ambientes de desenvolvimento, para gerir projetos com linguagens de programação variadas
- Para além de facilitar processos de gestão da compilação, gestão de versões entre outras, oferece mecanismos de gestão de dependências que facilita o processo de importação de bibliotecas essenciais aos projetos

Maven e JavaFX

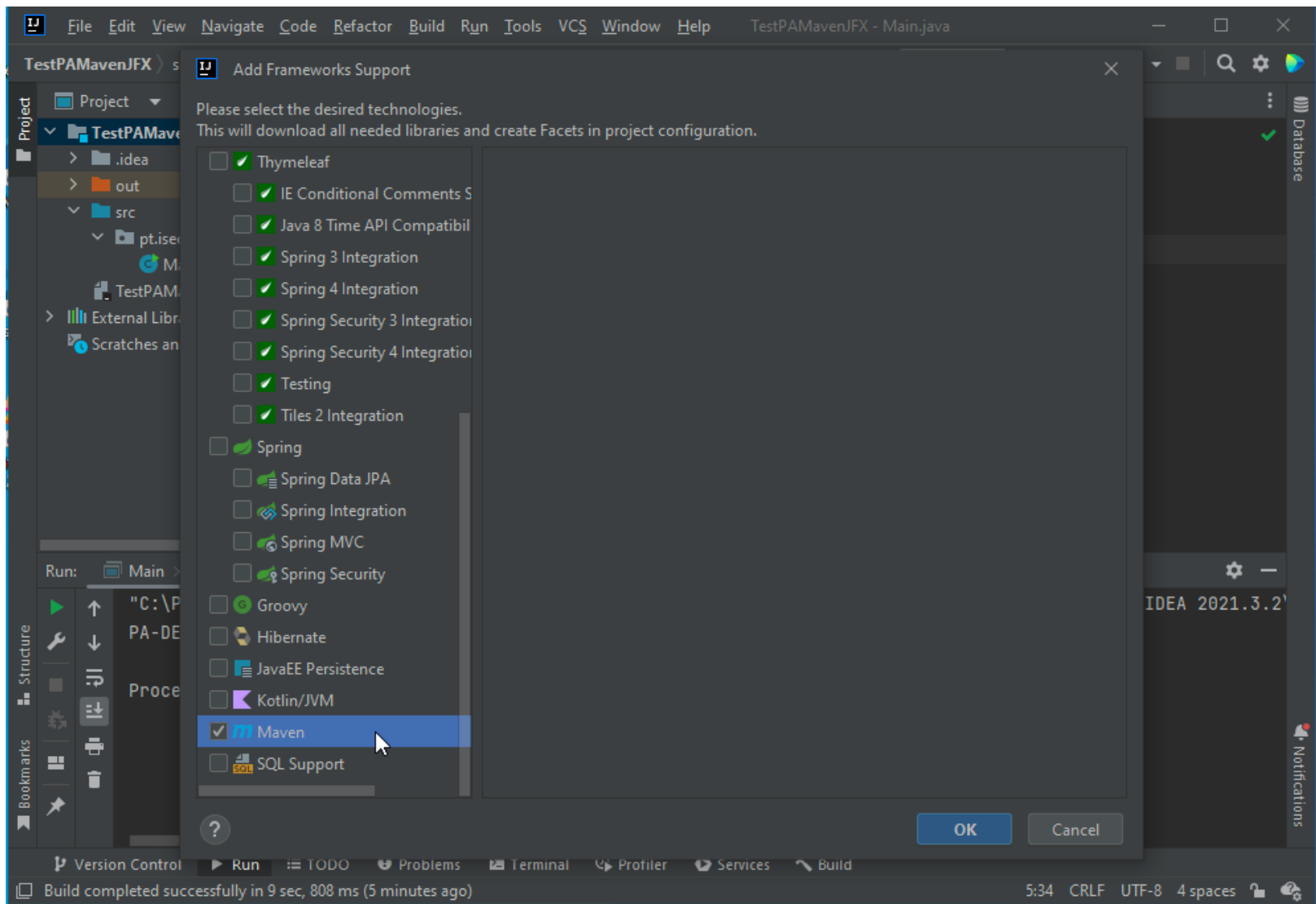
- Para incorporar o *JavaFX* num projecto já existente pode-se recorrer ao *Maven*, seguindo os seguintes passos:
 - Incorporar o *Maven* no projeto como *framework*
 - Configurar as informações básicas no ficheiro `pom.xml`
 - Configurar as dependências do projeto em relação às bibliotecas *JavaFX*
 - Incluir o *plugin* para compilação e execução de aplicações *JavaFX*
 - (Opcional) Fazer o *download* da documentação do *JavaFX* para facilitar o desenvolvimento

Incorporar o *Maven*

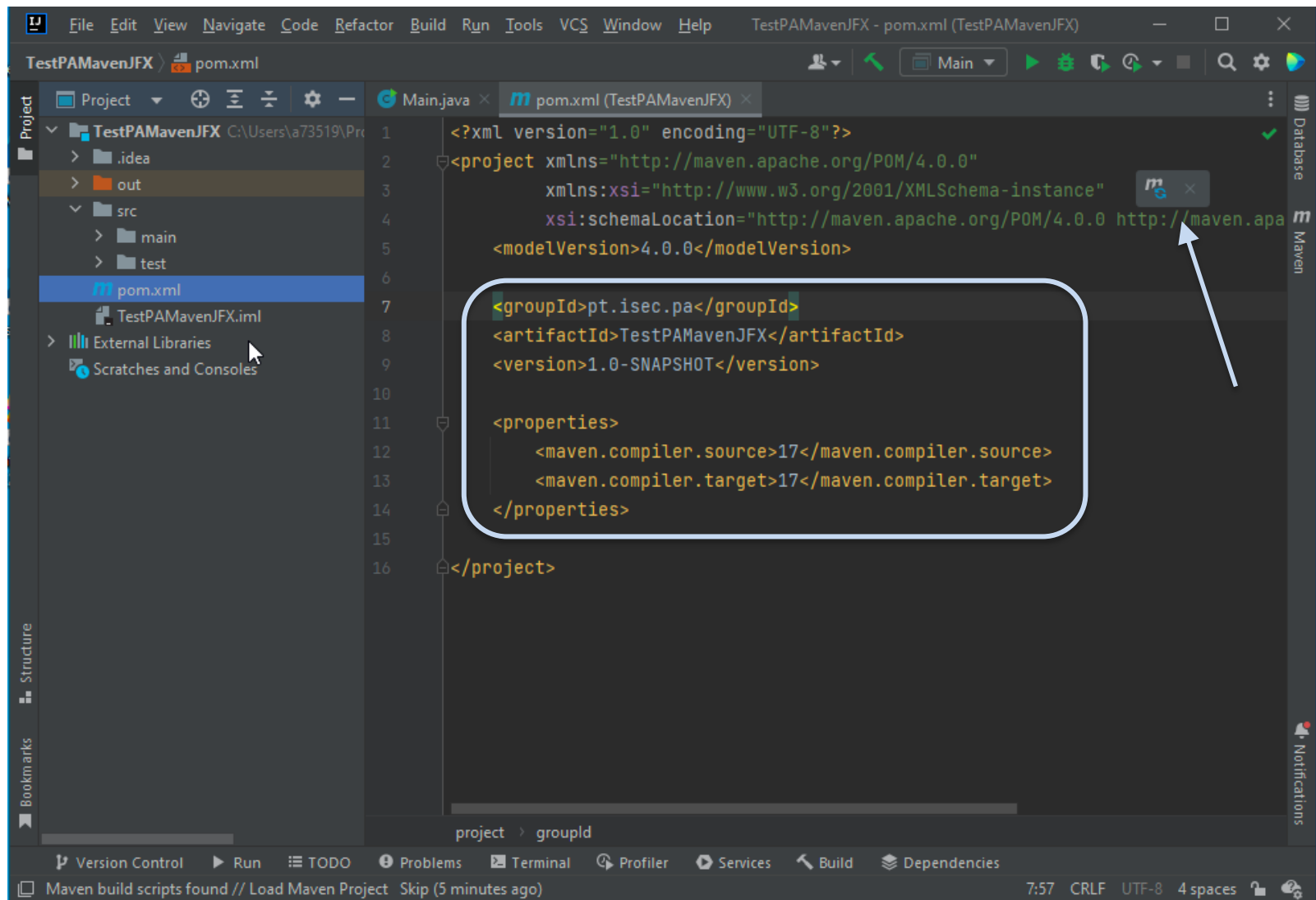
- Num projeto *Java* já existente ou criado para o efeito...
 - **Compilar e executar pelo menos uma vez**
 - Adicionar *Framework Maven* ao projeto



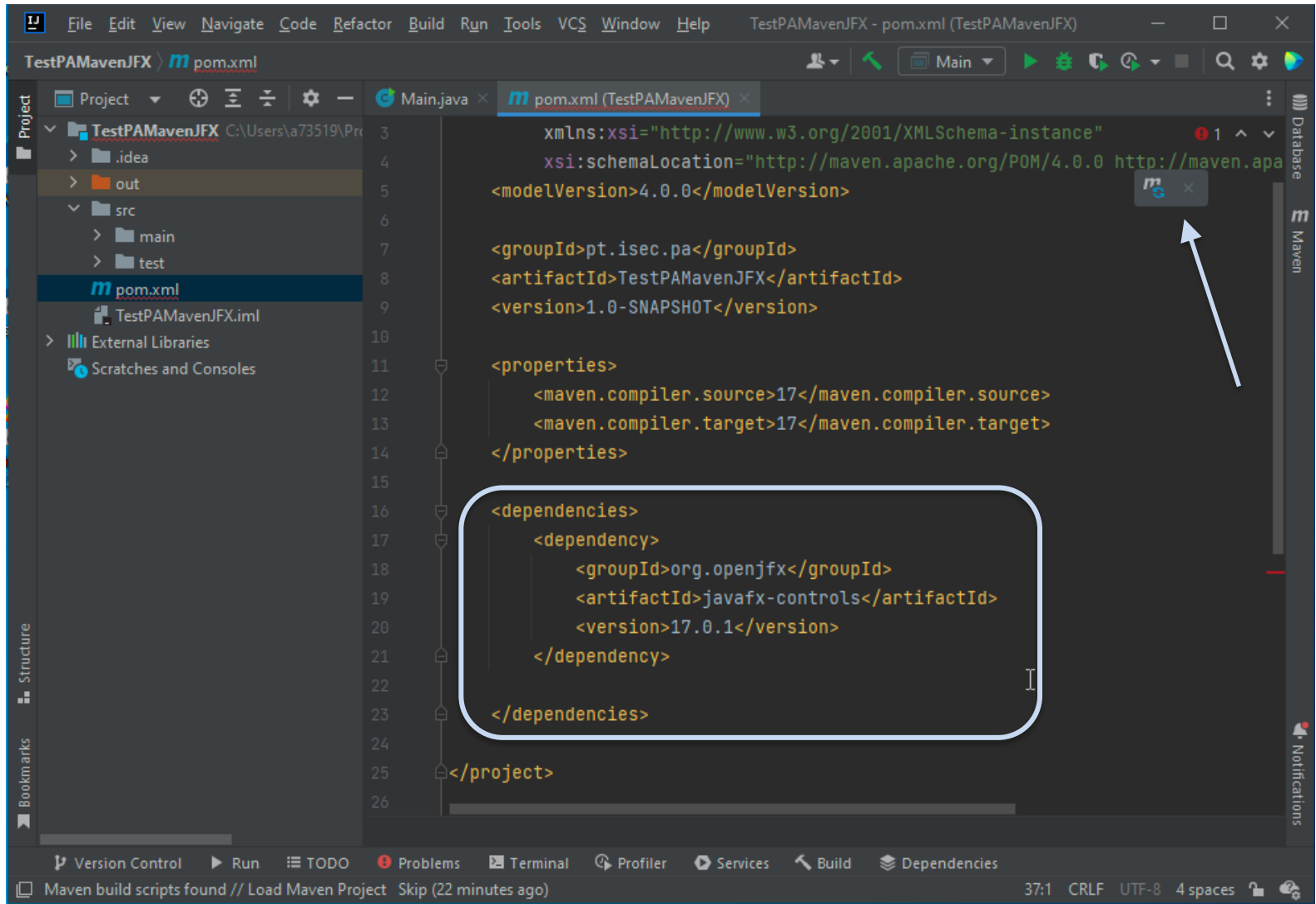
Incorporar o *Maven*



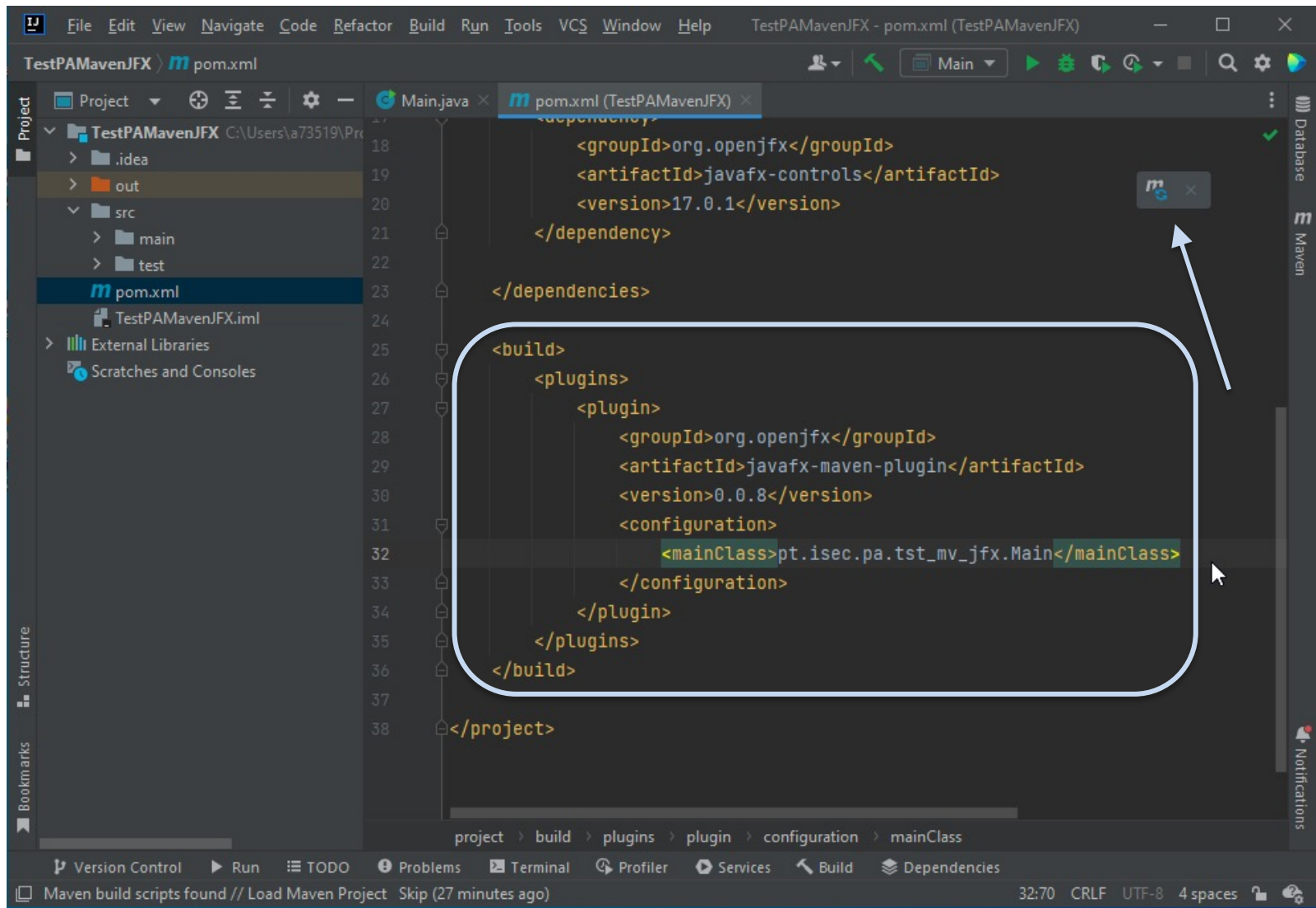
Incorporar o *Maven*



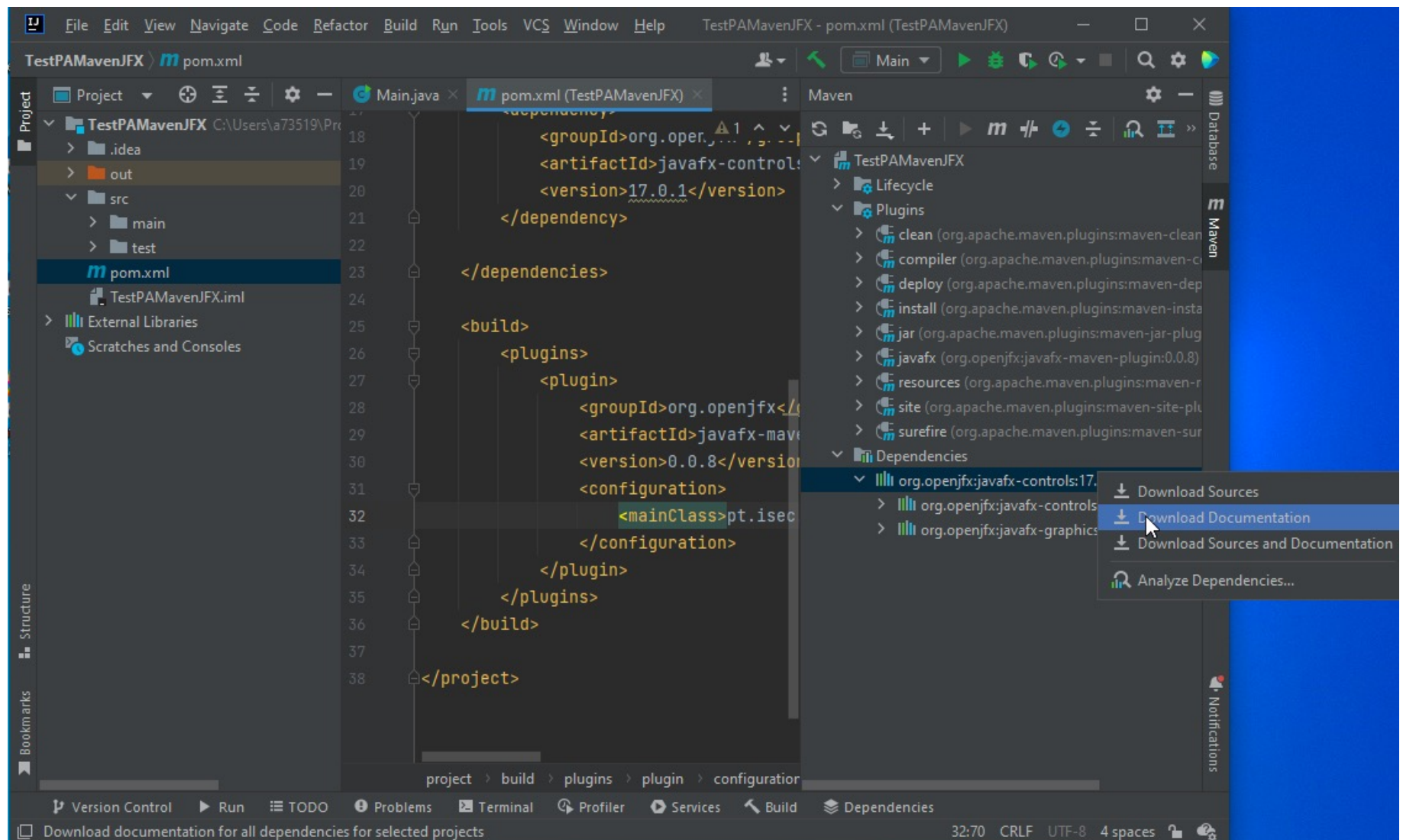
Configurar dependências



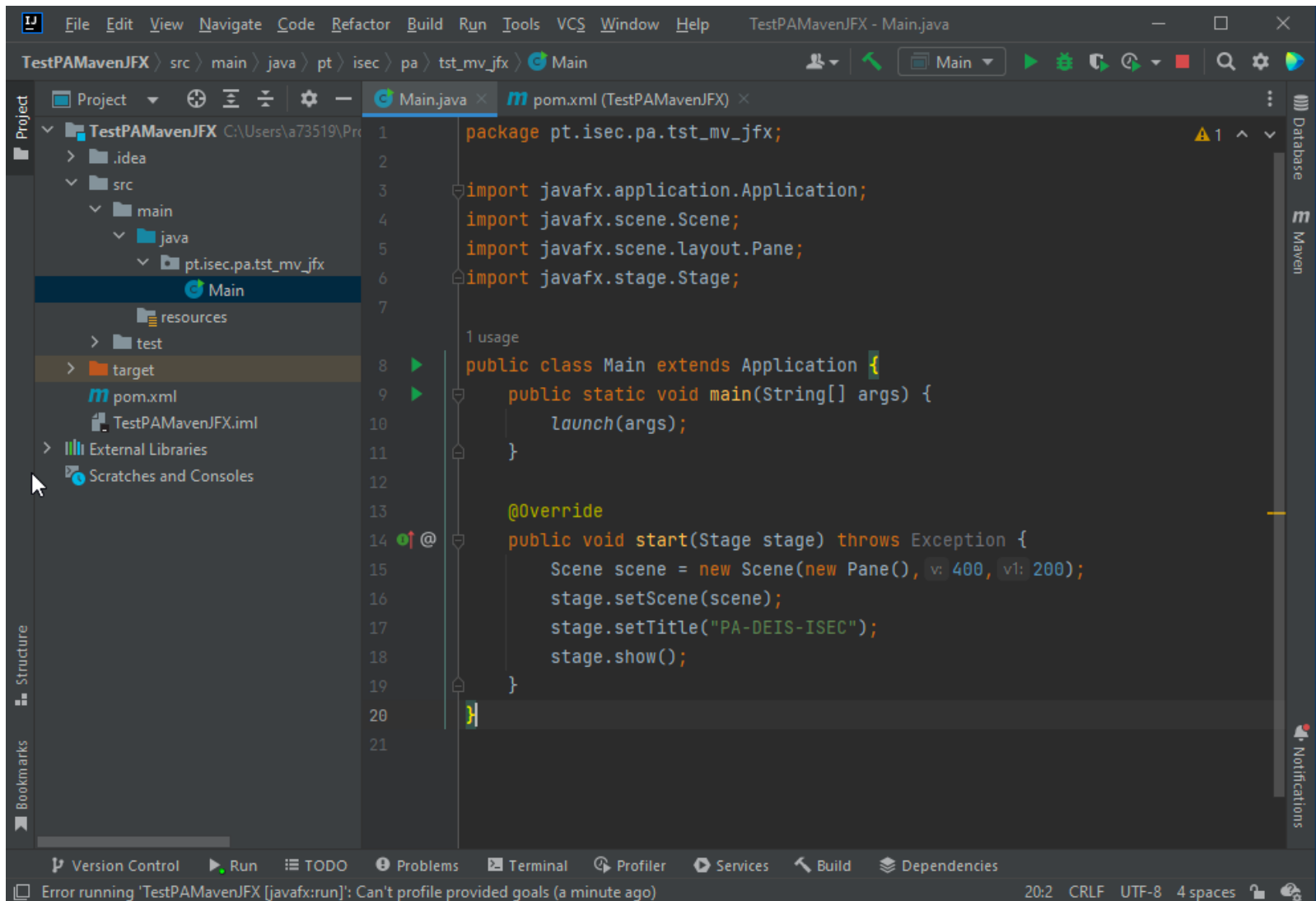
Incluir o *plugin* para JavaFX



Fazer *download* da documentação

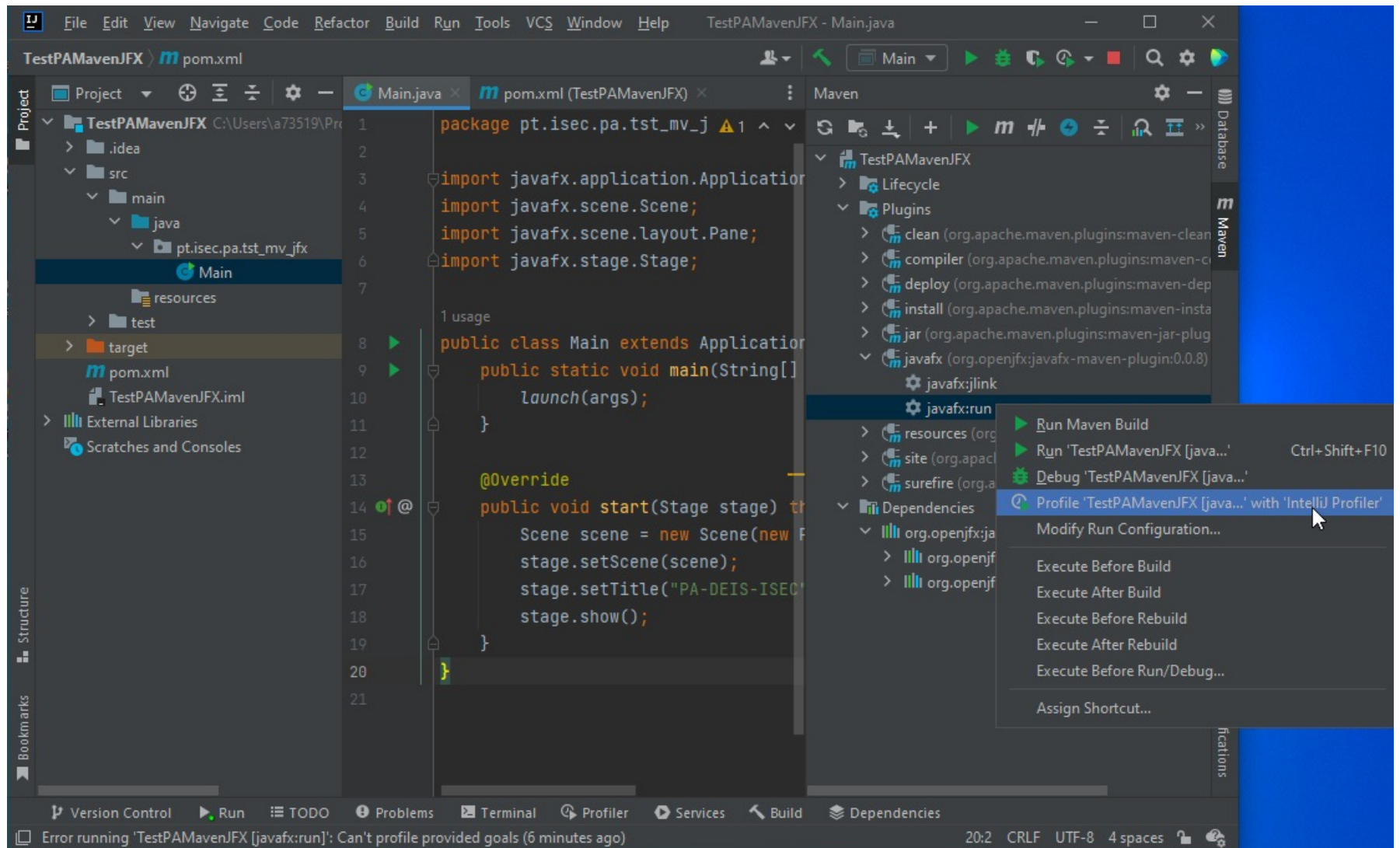


Experimental – Criar programa *JavaFX*

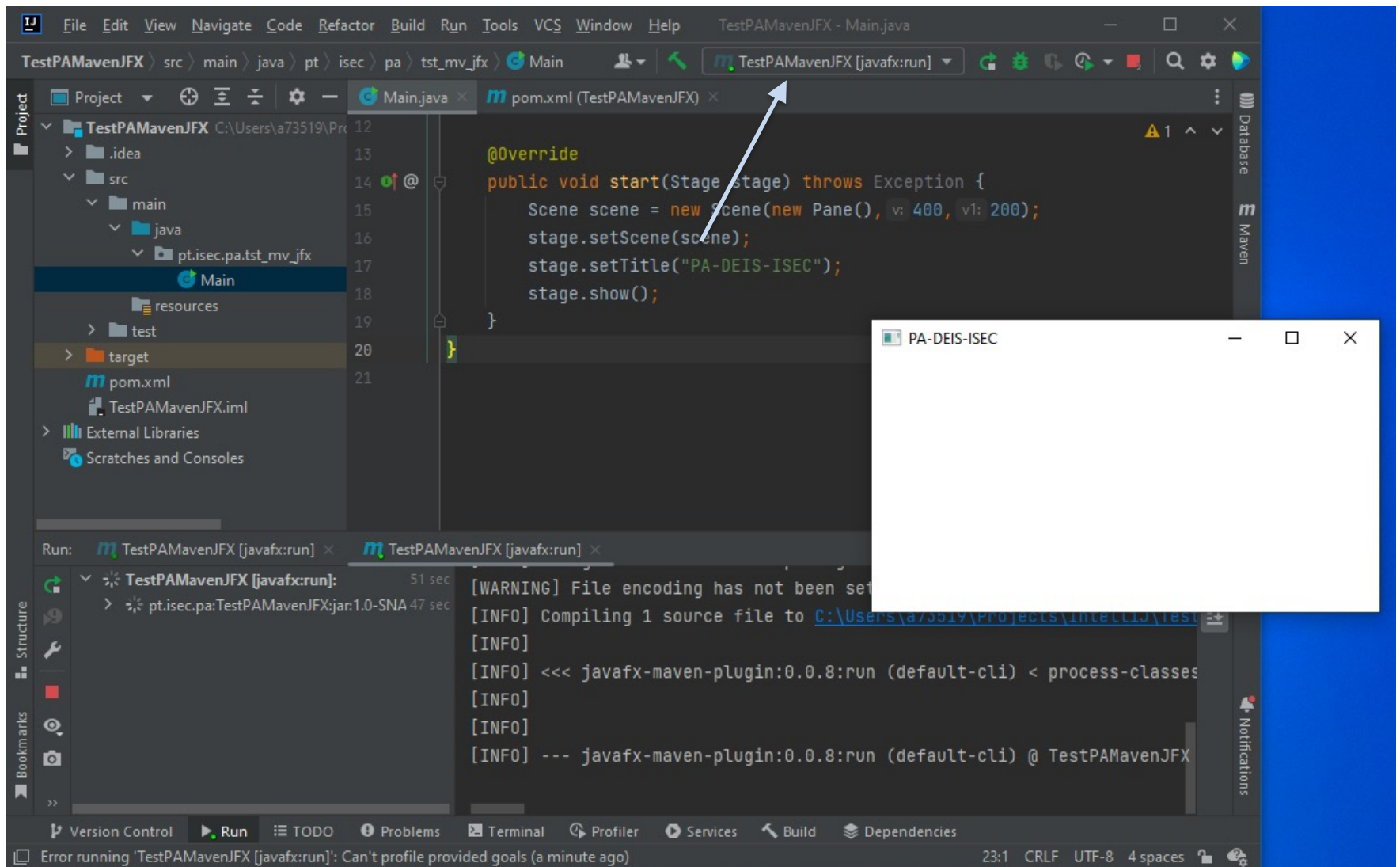


```
1 package pt.isec.pa.tst_mv_jfx;
2
3 import javafx.application.Application;
4 import javafx.scene.Scene;
5 import javafx.scene.layout.Pane;
6 import javafx.stage.Stage;
7
8 public class Main extends Application {
9     public static void main(String[] args) {
10         launch(args);
11     }
12
13     @Override
14     public void start(Stage stage) throws Exception {
15         Scene scene = new Scene(new Pane(), 400, 200);
16         stage.setScene(scene);
17         stage.setTitle("PA-DEIS-ISEC");
18         stage.show();
19     }
20 }
21
```

Experimental - Executar



Experimental - Executar



Experimental documentation

The screenshot shows an IDE window titled "TestPAMavenJFX - Main.java". The project structure on the left includes "TestPAMavenJFX" with subdirectories ".idea", "src", "resources", "test", and "target". The "src/main/java/pt.isec.pa.tst_mv_jfx" directory contains a "Main" class. The "Main.java" file is open, showing the following code:

```
1 usage
2
3 public class Main extends Application {
4     public static void main(String[] args) {
5         launch(args);
6     }
7
8     @Override
9     public void start(Stage stage) throws Exception {
10         Scene scene = new Scene(1000, 1000);
11         stage.setScene(scene);
12         stage.setTitle("TestPAMavenJFX");
13         stage.show();
14     }
15 }
```

A popup window displays the "Class Stage" hierarchy, showing that "Stage" extends "Window" and implements "EventTarget". The popup also includes a description of the "Stage" class: "The JavaFX Stage class is the top level JavaFX container. The primary Stage is constructed by the platform. Additional Stage objects may be constructed by the application. Stage objects must be constructed and modified on the JavaFX Application Thread. The JavaFX Application Thread is created as part of the startup process for the JavaFX runtime. See the Application class and the Platform.startup(Runnable) method for more."

The bottom of the IDE shows a "Run" tab with a successful build message: "[INFO] BUILD SUCCESS". The "Run" button is highlighted, and the "Run" configuration is set to "TestPAMavenJFX [javafx:run]".