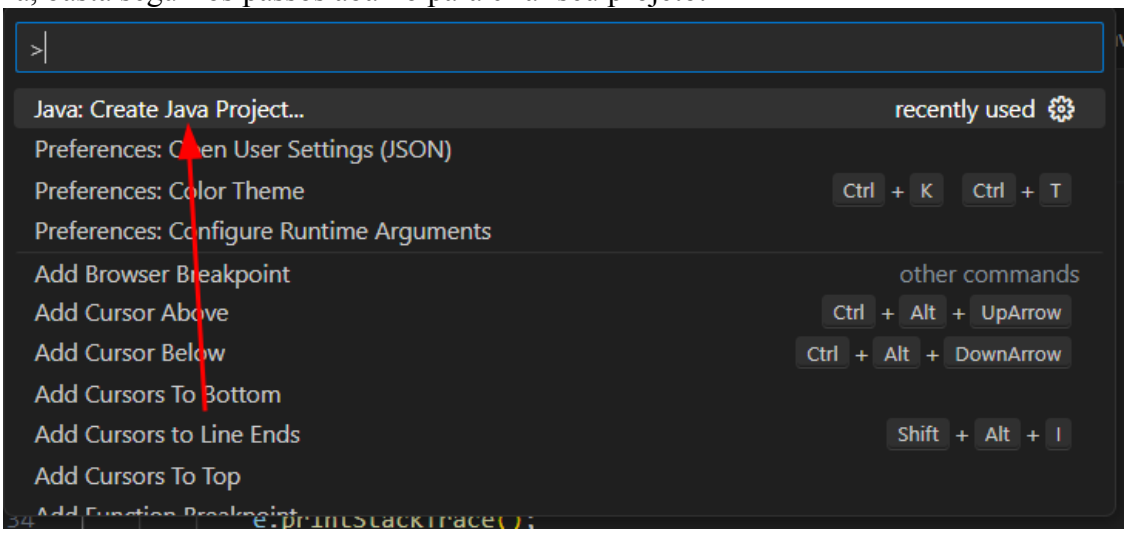
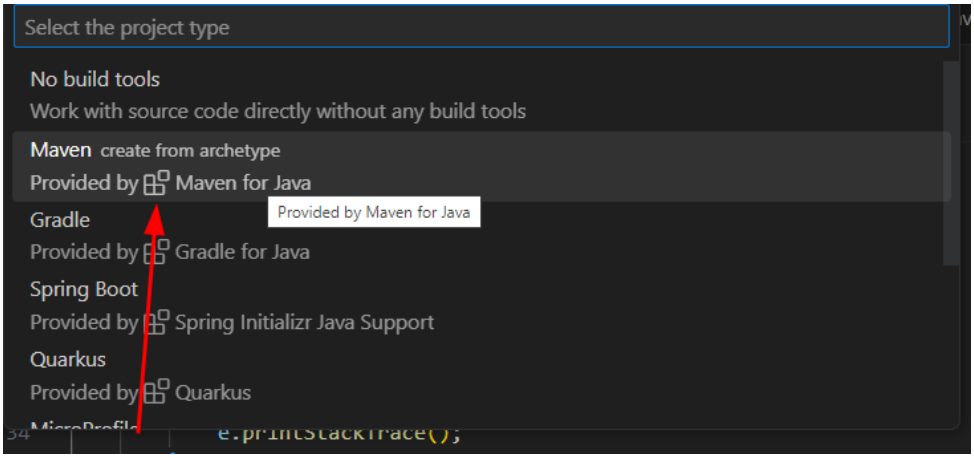
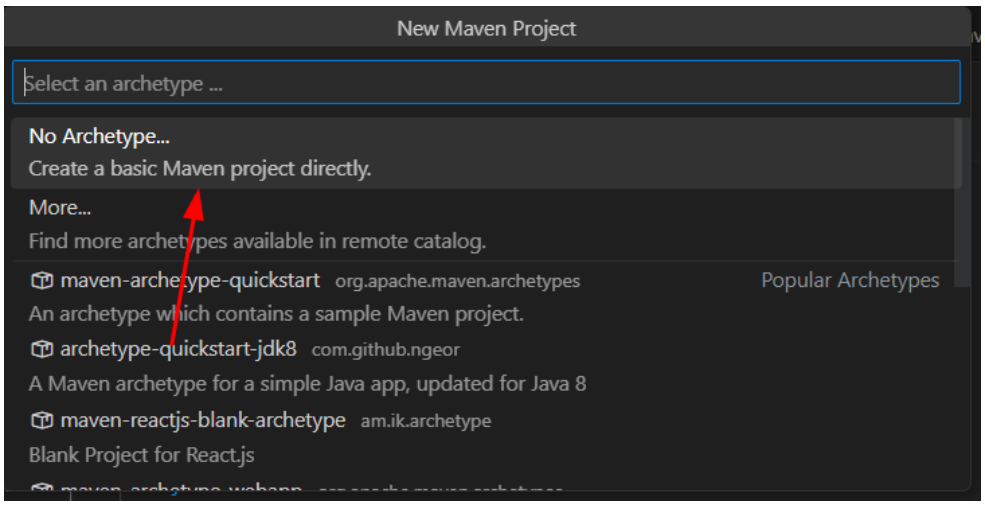
1. Instalar o Java: Você pode seguir os seguintes tutoriais a depender do seu sistema operacional:
Linux: https://youtu.be/QMeC_Ioin7g
Windows: https://youtu.be/sNFii-cvNz0

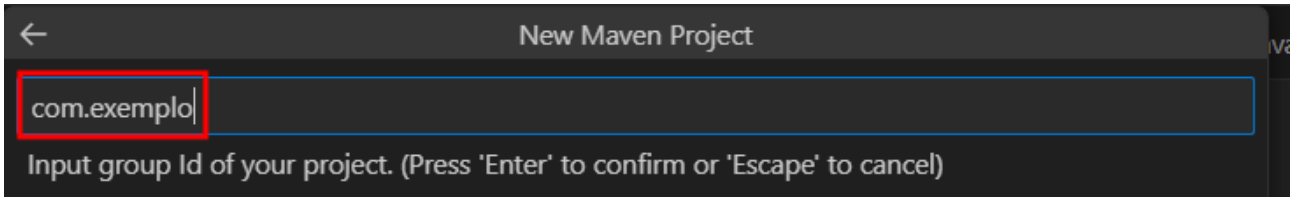
2. Instalar o Maven: Você pode seguir os seguintes tutoriais a depender do seu sistema operacional:
Linux: https://www.digitalocean.com/community/tutorials/install-maven-linux-ubuntu
Windows: https://youtu.be/x-VtjfGWc94

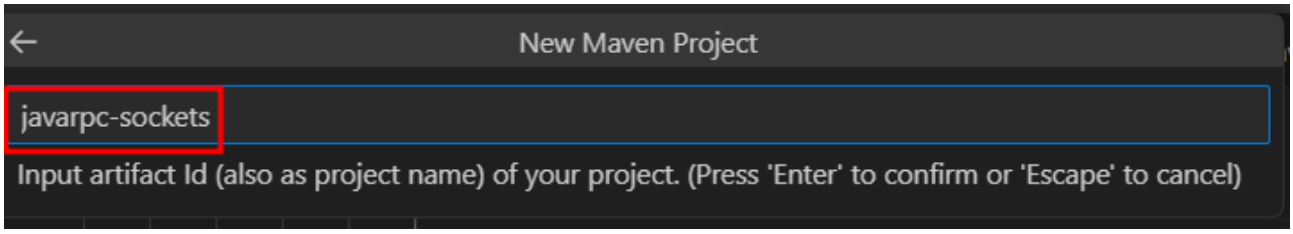
3. Criando o projeto: No VSCode, aperte Ctrl + Shift + P para abrir a barra de comandos. A partir de agora, basta seguir os passos abaixo para criar seu projeto:



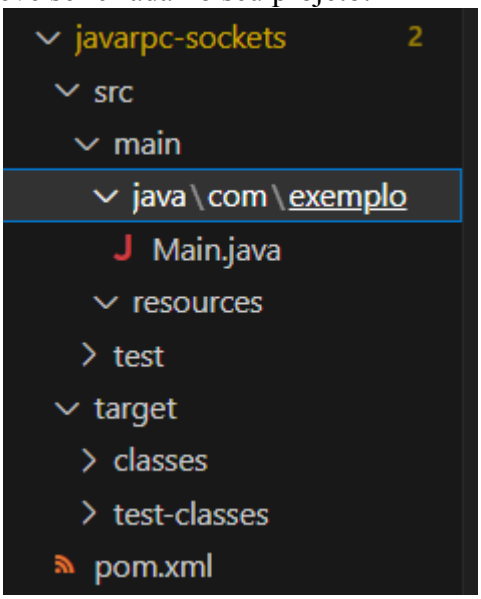








Uma estrutura semelhante a essa deve ser criada no seu projeto:



4. Configure o `pom.xml`: Navegue até o diretório do projeto e abra o arquivo `pom.xml`.

O `pom.xml` será bem básico:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>javarpc-sockets</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```

5. Criação do Código

Interface RPC
No diretório `src/main/java/com/exemplo`, crie a interface `HelloService.java`:

```
package com.exemplo;

// Interface do serviço
public interface HelloService {
   String sayHello(String name);
}
```

Implementação do Serviço

Crie a classe `HelloServiceImpl.java` que implementa a interface:

```
package com.exemplo;

// Implementação do serviço
public class HelloServiceImpl implements HelloService {
   @Override
   public String sayHello(String name) {
      return "Olá, " + name + "! Este é um exemplo de RPC com sockets.";
   }
}
```

Servidor

Crie a classe `Server.java`, responsável por aceitar conexões e processar as chamadas RPC:

```
package com.exemplo;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
   public static void main(String[] args) {
      try (ServerSocket serverSocket = new ServerSocket(5000)) {
         System.out.println("Servidor pronto na porta 5000...");

         // Espera por conexões
         while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("Cliente conectado!");

            try (BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                 PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

               // Recebe a requisição
               String input = in.readLine();
```

```
                System.out.println("Requisição recebida: " + input);

                // Processa a chamada ao método
                if (input.startsWith("sayHello:")) {
                    String name = input.split(":")[1];
                    HelloService service = new HelloServiceImpl();
                    String response = service.sayHello(name);

                    // Envia a resposta de volta ao cliente
                    out.println(response);
                } else {
                    out.println("Método não suportado.");
                }
            }
            clientSocket.close();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

Cliente

Crie a classe `Client.java`, que envia as chamadas RPC para o servidor:

```
package com.exemplo;

import java.io.*;
import java.net.Socket;

public class Client {
   public static void main(String[] args) {
      try (Socket socket = new Socket("localhost", 5000);
           BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
           PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

          // Envia a chamada do método para o servidor
          String request = "sayHello:Usuário";
          out.println(request);
          System.out.println("Requisição enviada: " + request);

          // Recebe a resposta do servidor
          String response = in.readLine();
          System.out.println("Resposta do servidor: " + response);

      } catch (IOException e) {
          e.printStackTrace();
      }
   }
}
```

6. Executando o código:
Abra o cmd ou terminal na pasta do seu projeto e siga os seguintes passos:

Agora podemos compilar nosso código: `mvn compile`

```
PS C:\Users\                              javarpc-sockets> mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.example:javarpc-sockets >--------------------
[INFO] Building javarpc-sockets 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ javarpc-sockets ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ javarpc-sockets ---
[INFO] Recompiling the module because of added or removed source files.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target\classes
[WARNING] bootstrap class path is not set in conjunction with -source 8
  not setting the bootstrap class path may lead to class files that cannot run on JDK 8
    --release 8 is recommended instead of -source 8 -target 1.8 because it sets the bootstrap class path automati
cally
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.605 s
[INFO] Finished at: 2025-05-03T12:26:56-03:00
[INFO] ------------------------------------------------------------------------
```

Agora execute o servidor com o seguinte comando: `java -cp target/classes com.exemplo.Server`

```
PS                                        \javarpc-sockets> java -cp target/classes com.exemplo.Server
Servidor pronto na porta 5000...
```

Em outro terminal, com o servidor ainda em execução, digite: `java -cp target/classes com.exemplo.Client`

```
                                          \javarpc-sockets> java -cp target/classes com.exemplo.Client
Requisição enviada: sayHello:2025.1- SD - 01A
Resposta do servidor: Olá, 2025.1- SD - 01A! Este é um exemplo de RPC com sockets.
```

Voltando ao servidor podemos ver as nossas requisições nos logs do servidor:

```
                                          \javarpc-sockets> java -cp target/classes com.exemplo.Server
Servidor pronto na porta 5000...
Cliente conectado!
Requisição recebida: sayHello:2025.1- SD - 01A
```