



# Sejam bem-vindos!

## PROGRAMAÇÃO ORIENTADA A OBJETOS

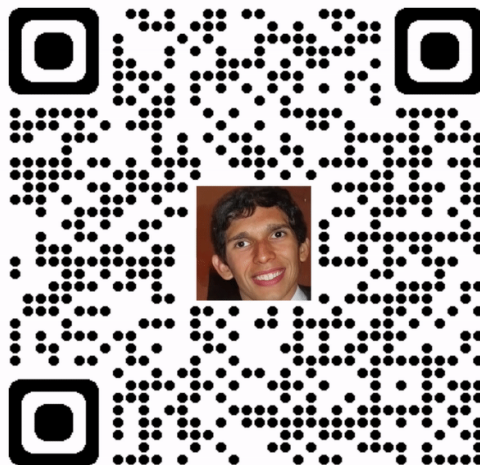


# Diogenes Carvalho Matias


Formação:

- **Graduação: Sistemas de Informação;**
- **Especialista em: Engenharia e Arquitetura de Software;**
- **MBA EXECUTIVO EM BUSINESS INTELLIGENCE (em andamento);**
- **Mestrado Acadêmico em Engenharia de Computação (UPE em andamento);**

Maiores informações :



[Linkedin](#)



Vamos montar nosso ambiente para conexão com o banco de dados!

OBS: Caso tenha o MySQL instalado, não precisa instalar o ambiente a seguir!



O XAMPP é o ambiente de desenvolvimento PHP mais popular

O XAMPP é completamente gratuito, de fácil de instalar a distribuição Apache, contendo MySQL, PHP e Perl. O pacote de código aberto do XAMPP foi criado para ser extremamente fácil de instalar e de usar.

<https://www.youtube-nocookie.com/embed/h6DEDm7C37A>



**XAMPP Control Panel v3.2.4**

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	2136 6340	80, 443	<b>Stop</b> Admin Config Logs
<input type="checkbox"/>	MySQL	4216	3306	<b>Stop</b> Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

23:54:37 [main] Initializing Control Panel  
23:54:37 [main] Windows Version: Home 64-bit  
23:54:37 [main] XAMPP Version: 7.3.10  
23:54:37 [main] Control Panel Version: 3.2.4 [ Compiled: Jun 5th 2019 ]  
23:54:37 [main] You are not running with administrator rights! This will work for most application stuff but whenever you do something with services there will be a security dialogue or things will break! So think about running this application with administrator rights!  
23:54:37 [main] XAMPP Installation Directory: "c:\xampp1\  
23:54:37 [main] Checking for prerequisites  
23:54:37 [main] All prerequisites found  
23:54:37 [main] Initializing Modules  
23:54:37 [main] Starting Check-Timer  
23:54:37 [main] Control Panel Ready  
23:54:41 [Apache] Attempting to start Apache app...  
23:54:41 [Apache] Status change detected: running  
23:54:43 [mysql] Attempting to start MySQL app...  
23:54:43 [mysql] Status change detected: running

O XAMPP com nosso servidor instalado vamos inicializa-lo.

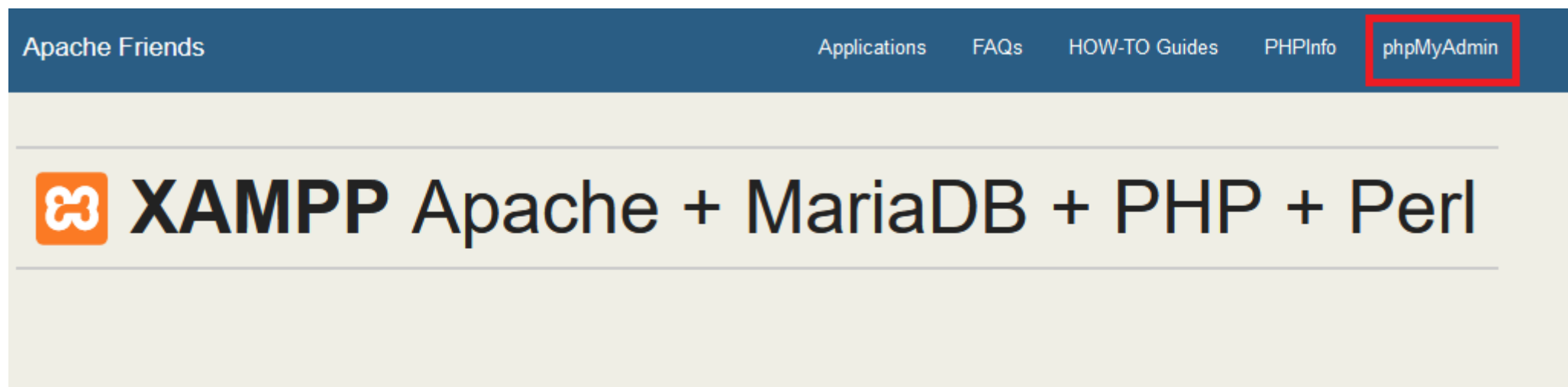


O vamos acessar nosso MySQL pelo servidor XAMPP, basta escolher um navegador de internet agora e digitar:

<http://localhost/dashboard/>



O vamos escolher a opção **phpMyAdmin**:



## Welcome to XAMPP for Windows 7.3.10

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the [FAQs](#) section or check the [HOW-TO Guides](#) for getting started with PHP applications.



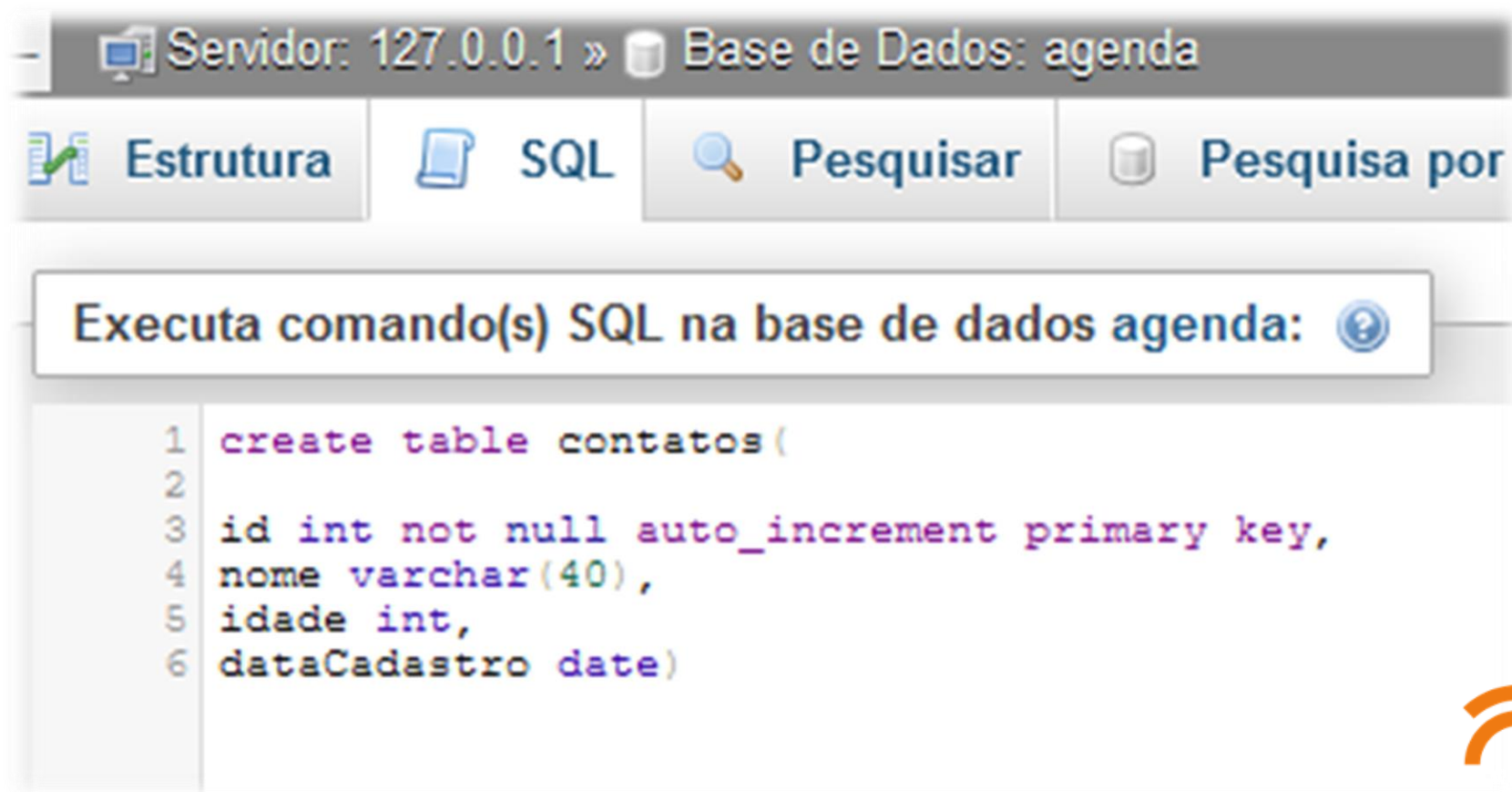
O vamos escolher a opção phpMyAdmin->Banco de dados  
vamos criar uma nova base de dados chamado : **agenda**



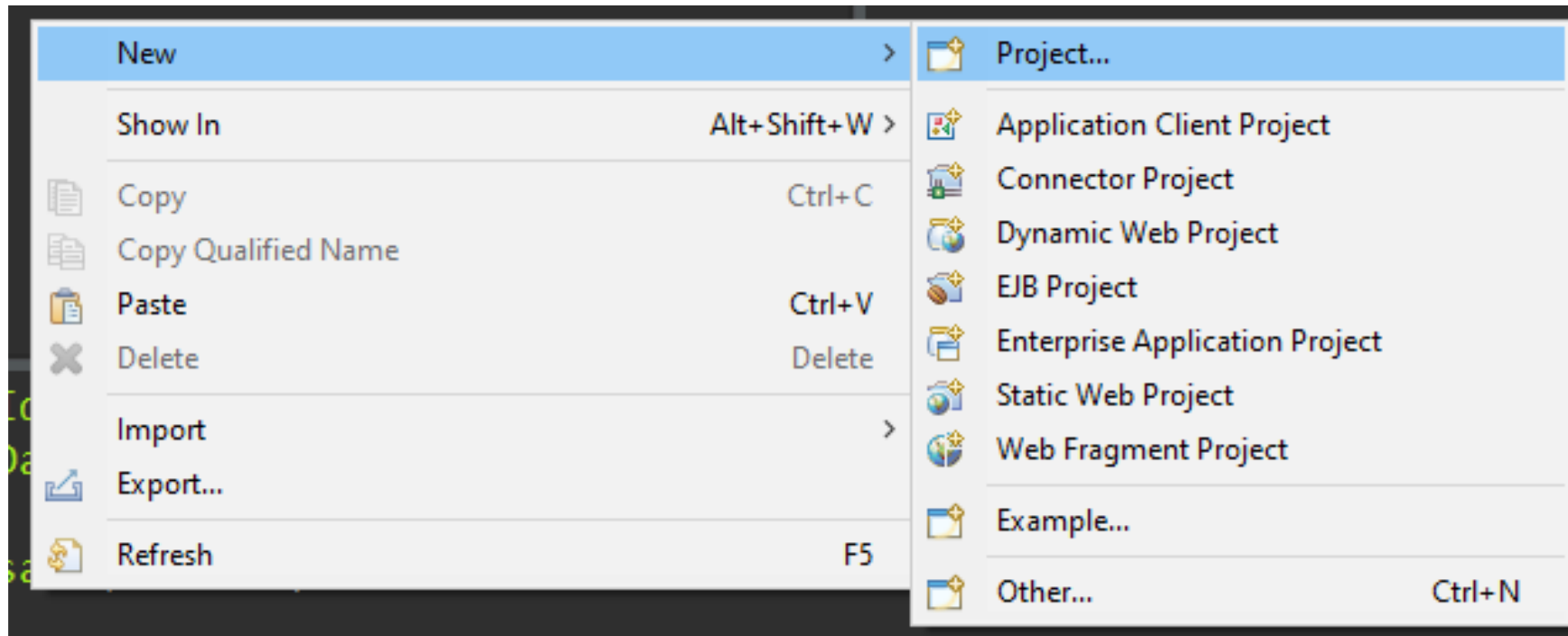




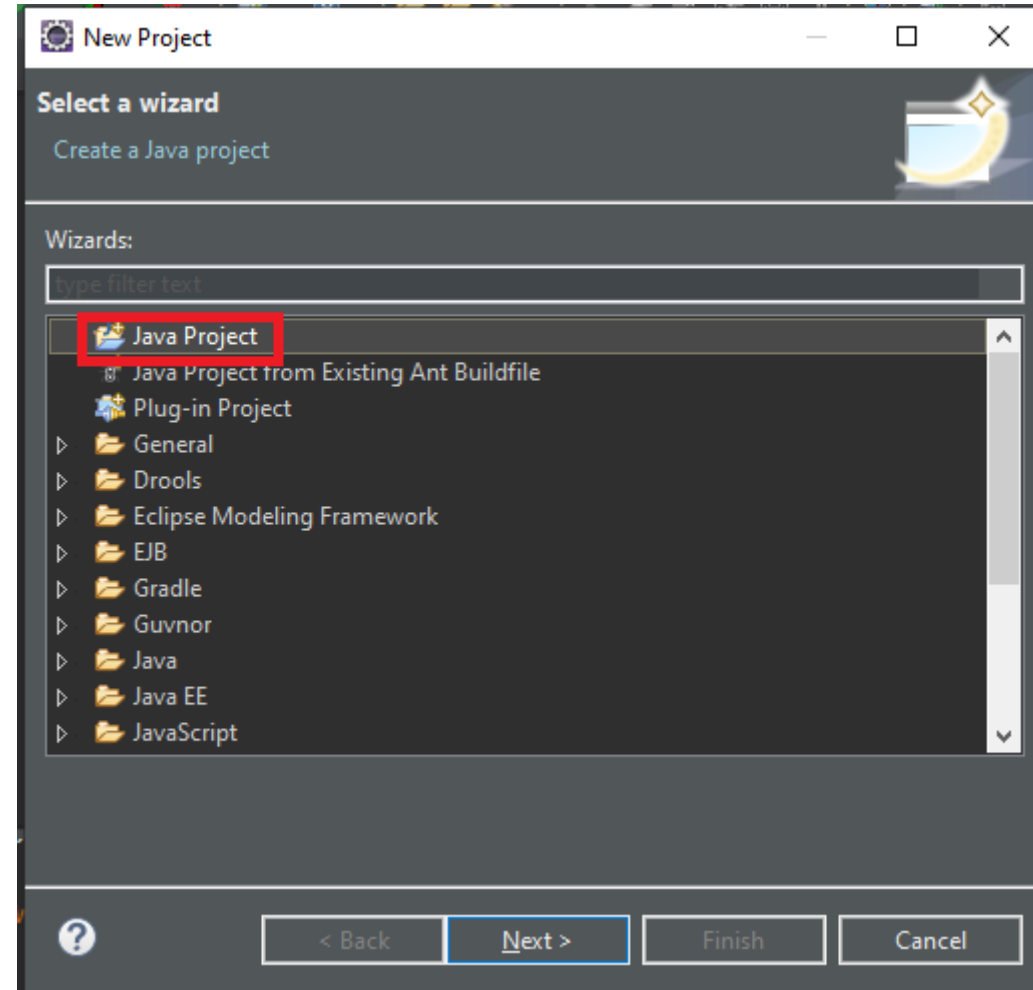
O com nossa base de dados Agenda já criada vamos criar nossa tabela chamada Contato e logo após clicar no botão Executar.



Pronto nossa base de dados foi criado agora vamos criar um projeto Java e importar o conector de nossa base de dados MySQL para que nosso projeto possa se conectar com o banco de dados.



Vamos escolher Java Project-> Next



## Create a Java Project

Create a Java project in the workspace or in an external location.



Project name: POOBanco

☒ Use default location

Location: C:\Users\Dcmaster\eclipse-workspace\POOBanco1

Browse...

## JRE

☐ Use an execution environment JRE:

JavaSE-1.8

☐ Use a project specific JRE:

jre1.8.0\_221

☐ Use default JRE (currently 'jre1.8.0\_221')[Configure JREs...](#)

## Project layout

☐ Use project folder as root for sources and class files☐ Create separate folders for sources and class files[Configure default...](#)

## Working sets

☐ Add project to working sets

New...

Working sets:

Select...



&lt; Back

Next &gt;

Finish

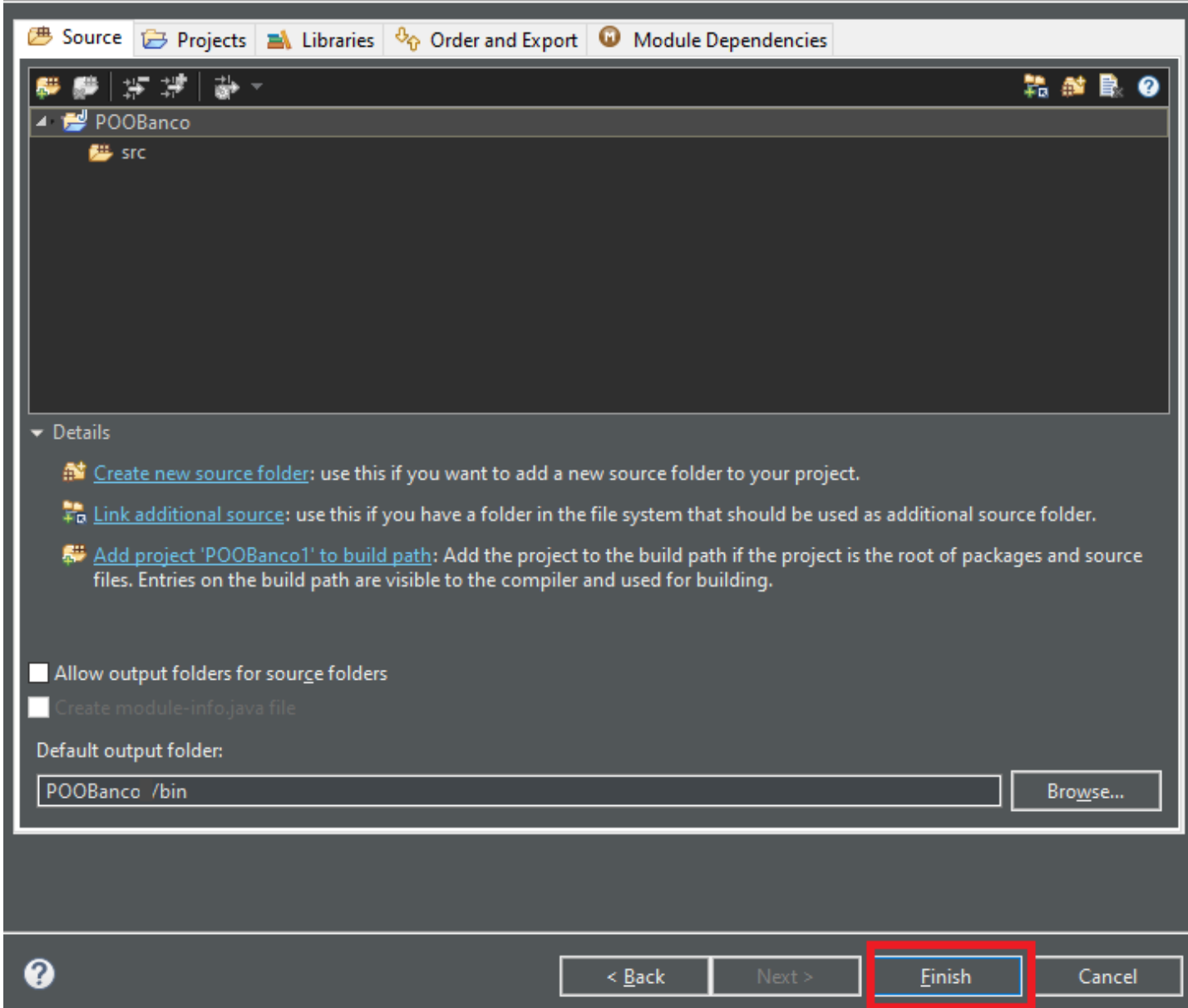
Cancel

Vamos dar um nome POOBANCO  
Ou um nome a sua escolha.



## Java Settings

Define the Java build settings.



Source Projects Libraries Order and Export Module Dependencies

POOBanco

src

Details

- [Create new source folder](#): use this if you want to add a new source folder to your project.
- [Link additional source](#): use this if you have a folder in the file system that should be used as additional source folder.
- [Add project 'POOBanco1' to build path](#): Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

☐ Allow output folders for source folders

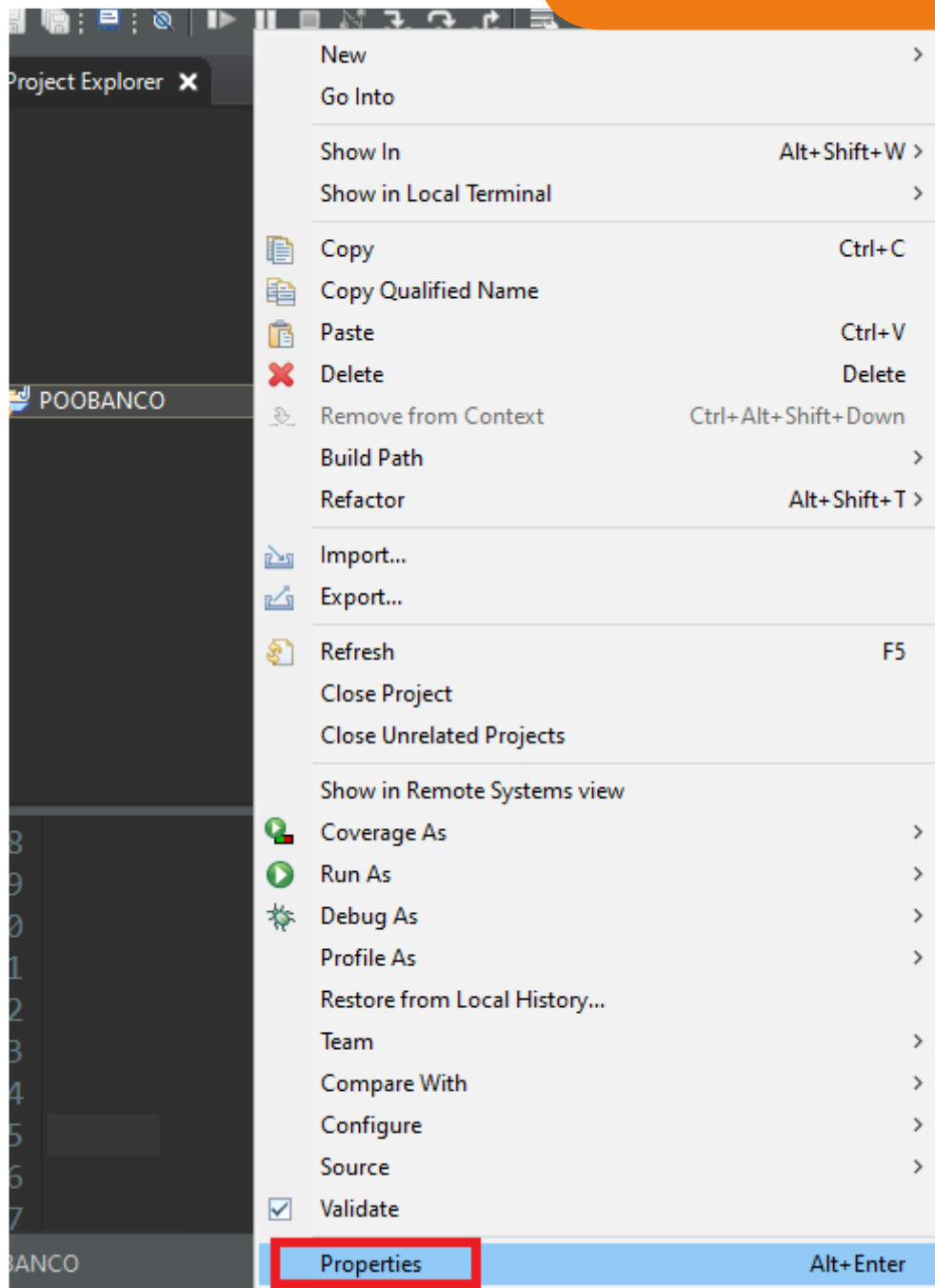
☐ Create module-info.java file

Default output folder:

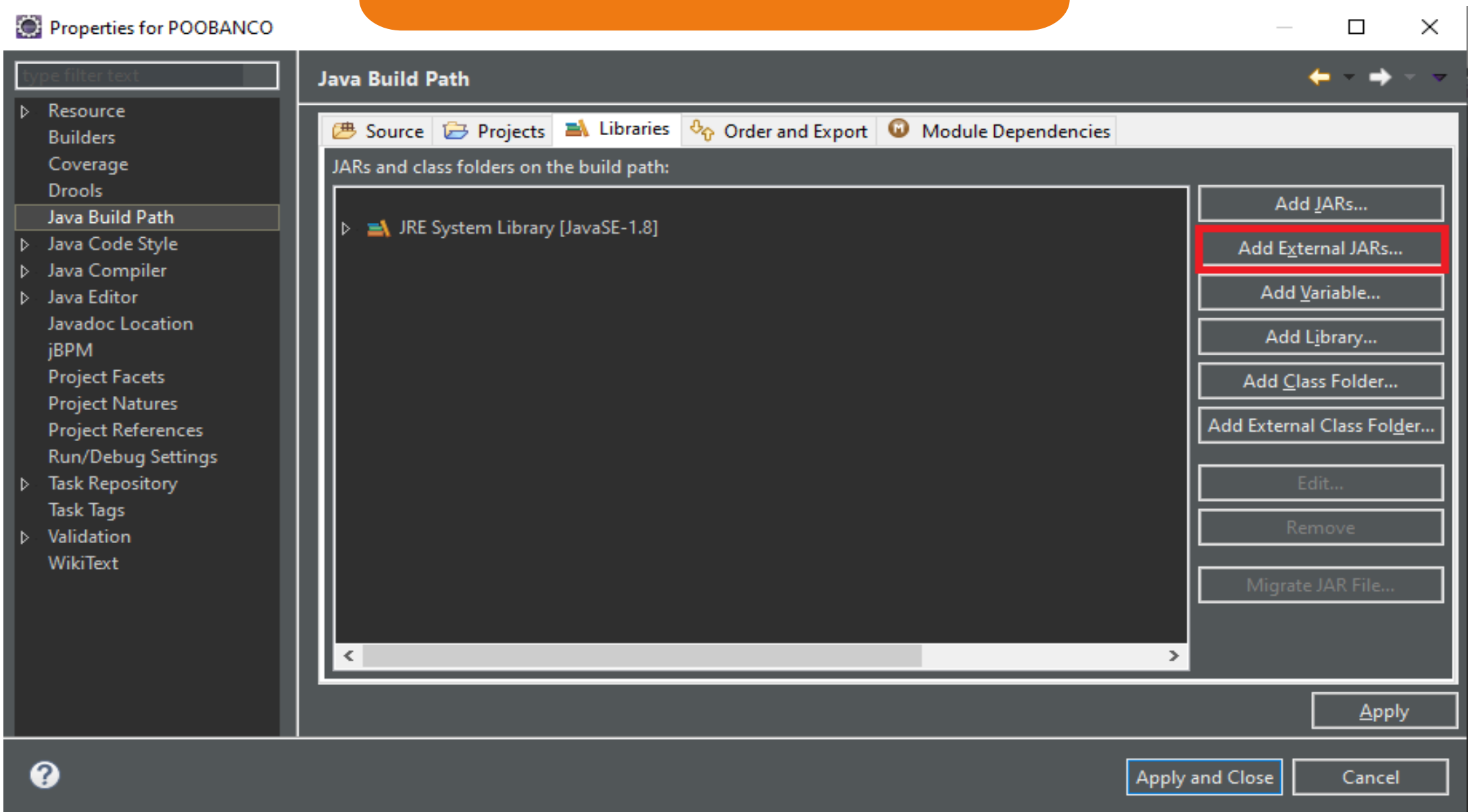
POOBanco /bin Browse...

< Back Next > **Finish** Cancel

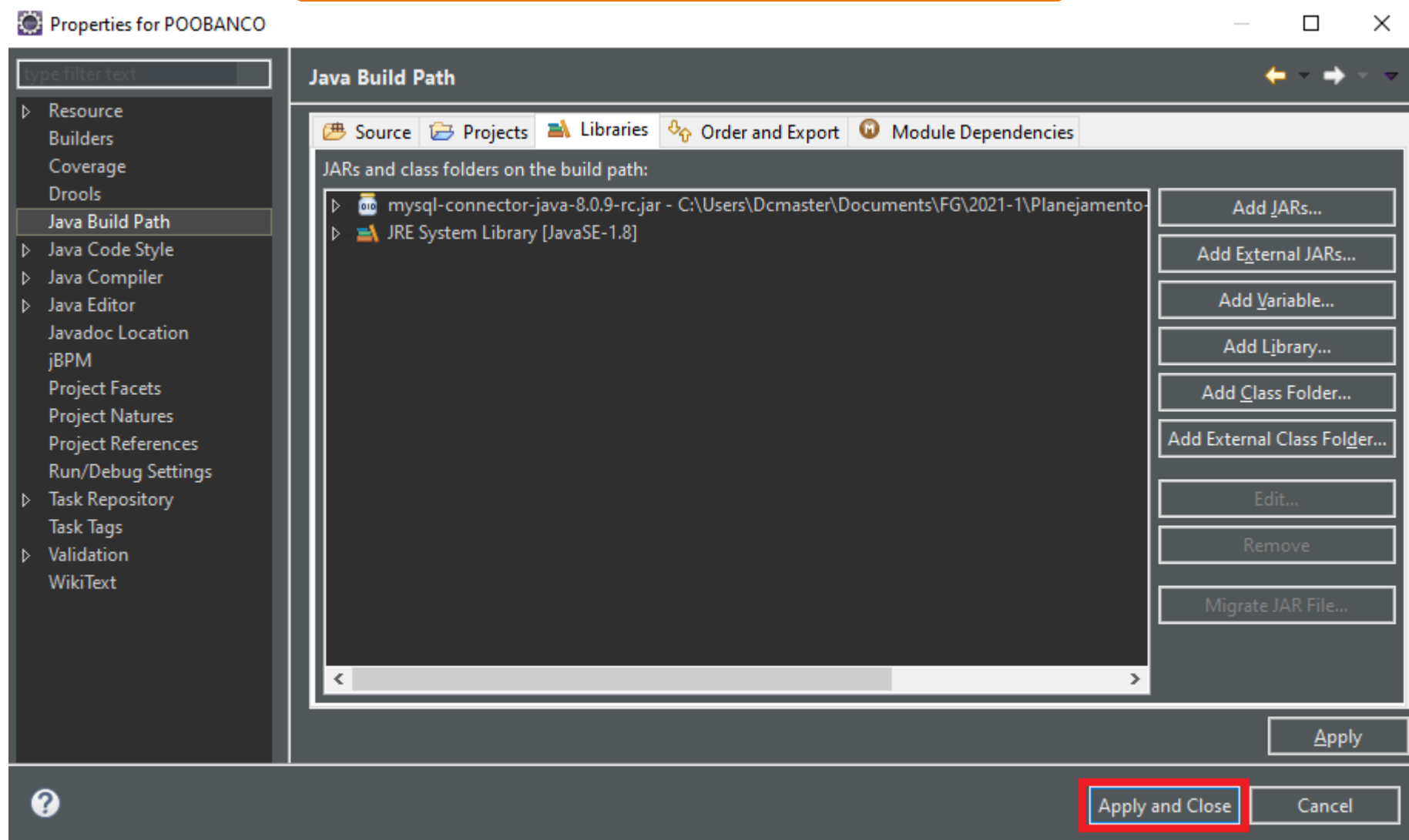
Só finalizar a construção do nosso projeto Finish.



Vamos importar o nosso conector em nosso projeto criado, clicar com o botão direito em nosso projeto e Ir em propriedades.



Vamos pegar nosso conector do mysql .jar



Agora só Apply and Close



Vamos testar se nossa conexão esta funcionando:

Crie uma classe em nosso projeto chamado: **ConnectionFactory**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.TimeZone;

public class ConnectionFactory {

    private static final String URL="jdbc:mysql://localhost:3306/agenda?serverTimezone="+TimeZone.getDefault().getID();
    private static final String DRIVER="com.mysql.cj.jdbc.Driver";
    private static final String USUARIO="root";
    private static final String SENHA="";

    public static Connection getConnection() throws SQLException{
        try{
            Class.forName(DRIVER);
            System.out.println("Conectado ao banco");
            return DriverManager.getConnection(URL, USUARIO, SENHA);
        }
        catch (ClassNotFoundException e){
            throw new SQLException(e.getMessage());
        }
    }
}
```

Vamos testar se nossa conexão esta funcionando:

Crie uma classe em nosso projeto chamado: Principal

```
import java.sql.Connection;

public class Princiapl {

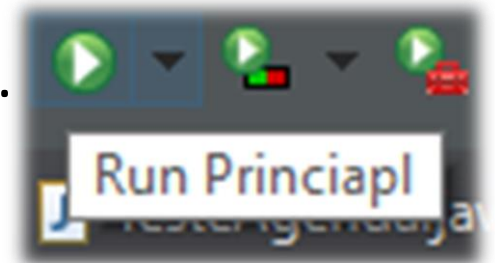
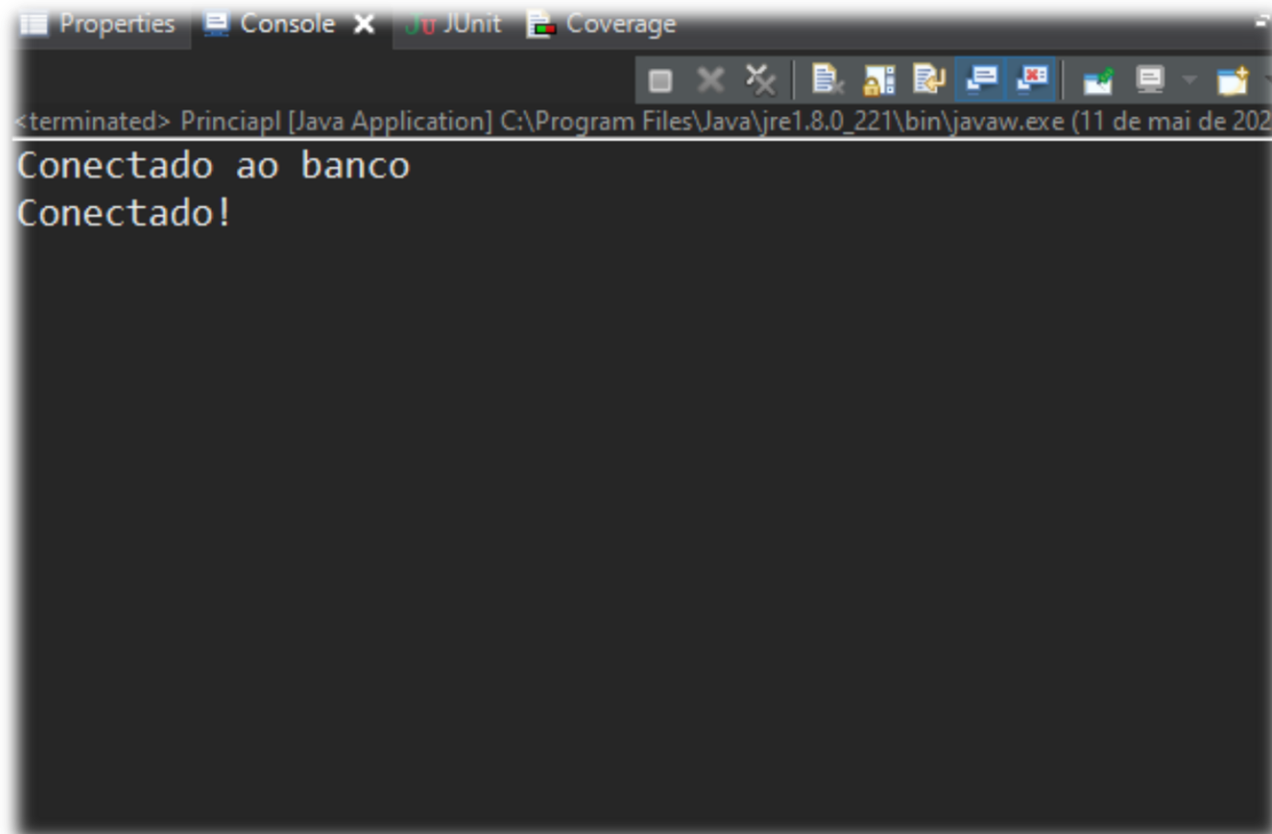
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try{
            Connection con = ConnectionFactory.getConnection();
            System.out.println("Conectado!");
            con.close();
        }catch (java.sql.SQLException e) {
            e.printStackTrace();
        }

    }

}
```

Vamos testar se nossa conexão esta funcionando:

Vamos testar o projeto para ver se conectou com o baco de dados.



Agora com tudo funcionando perfeitamente vamos criar nossas classe para inserir os dados no banco de dados:

- 1- Contato;
- 2- ContatoDAO;
- 3- TesteAgenda;

```
import java.util.Date;

public class Contato {

    private int id;
    private String nome;
    private int idade;
    private Date dataCadastro;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }
}
```

## Vamos criar nossa Classe Contato Parte-01

```
public void setNome(String nome) {  
    this.nome = nome;  
}
```

```
public int getIdade() {  
    return idade;  
}
```

```
public void setIdade(int idade) {  
    this.idade = idade;  
}
```

```
public Date getDataCadastro() {  
    return dataCadastro;  
}
```

```
public void setDataCadastro(Date dataCadastro) {  
    this.dataCadastro = dataCadastro;  
}
```

## Vamos criar nossa Classe Contato Parte-02

```
}
```

# Vamos criar nossa Classe ContatoDAO

## Parte-01

```
import java.sql.Connection;  
import java.sql.Date;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;
```

```
import agenda.ConnectionFactory;  
import agenda.Contato;
```

```
public class ContatoDAO {  
    public void save(Contato contato) {
```

```
        String sql = "INSERT INTO contatos(nome,idade,dataCadastro)" + " VALUES(?,?,?)";
```

```
        Connection conn = null;
```

```
        PreparedStatement pstm = null;
```

```
try {  
    // Cria uma conexão com o banco  
    conn = ConnectionFactory.getConnection();  
    // Cria um PreparedStatement, classe usada para executar a query  
    pstmt = conn.prepareStatement(sql);  
    // Adiciona o valor do primeiro parâmetro da sql  
    pstmt.setString(1, contato.getNome());  
    // Adicionar o valor do segundo parâmetro da sql  
    pstmt.setInt(2, contato.getIdade());  
    // Adiciona o valor do terceiro parâmetro da sql  
    pstmt.setDate(3, new Date(contato.getDataCadastro().getTime()));  
    // Executa a sql para inserção dos dados  
    pstmt.execute();  
} catch (Exception e) {  
    e.printStackTrace();  
}  
finally {  
    // Fecha as conexões  
    try {
```

## Vamos criar nossa Classe ContatoDAO Parte-02



```
        if (pstm != null) {
            pstm.close();
        }
        if (conn != null) {
            conn.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

Vamos criar nossa Classe ContatoDAO Parte-03

```
public void removeById(int id) {
    String sql = "DELETE FROM contatos WHERE id = ?";
    Connection conn = null;
    PreparedStatement pstm = null;
    try {
        conn = ConnectionFactory.getConnection();
        pstm = conn.prepareStatement(sql);
        pstm.setInt(1, id);
        pstm.execute();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (pstm != null) {
                pstm.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (Exception e) {
```



```
        e.printStackTrace();  
    }  
}  
}
```

Vamos criar nossa Classe ContatoDAO Parte-05

```
public void update(Contato contato) {  
  
String sql = "UPDATE contatos SET nome = ?, idade = ?, dataCadastro = ?" + " WHERE id = ?";  
  
    Connection conn = null;  
    PreparedStatement pstmt = null;
```

## Vamos criar nossa Classe ContatoDAO Parte-06

```
try {
```

```
    // Cria uma conexão com o banco  
    conn = ConnectionFactory.getConnection();  
    // Cria um PreparedStatement, classe usada para executar a query  
    pstmt = conn.prepareStatement(sql);  
    // Adiciona o valor do primeiro parâmetro da sql  
    pstmt.setString(1, contato.getNome());  
    // Adicionar o valor do segundo parâmetro da sql  
    pstmt.setInt(2, contato.getIdade());  
    // Adiciona o valor do terceiro parâmetro da sql  
    pstmt.setDate(3, new Date(contato.getDataCadastro().getTime()));  
    pstmt.setInt(4, contato.getId());  
    // Executa a sql para inserção dos dados  
    pstmt.execute();
```

```
} catch (Exception e) {  
    e.printStackTrace();  
}  
finally {  
    // Fecha as conexões  
    try {
```

```
        if (pstmt != null) {
```

# Vamos criar nossa Classe ContatoDAO Parte-07



```
        pstmt.close();
    }
    if (conn != null) {
        conn.close();
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Vamos criar nossa Classe ContatoDAO Parte-08

```
public List<Contato> getContatos() {  
  
    String sql = "SELECT * FROM contatos";  
    List<Contato> contatos = new ArrayList<Contato>();  
    Connection conn = null;  
    PreparedStatement pstm = null;  
    // Classe que vai recuperar os dados do banco de dados  
    ResultSet rset = null;
```

## Vamos criar nossa Classe ContatoDAO Parte-09

```
try {  
    conn = ConnectionFactory.getConnection();  
    pstmt = conn.prepareStatement(sql);  
    rset = pstmt.executeQuery();  
    // Enquanto existir dados no banco de dados, faça  
    while (rset.next()) {  
        Contato contato = new Contato();  
        // Recupera o id do banco e atribui ele ao objeto  
        contato.setId(rset.getInt("id"));  
        // Recupera o nome do banco e atribui ele ao objeto  
        contato.setNome(rset.getString("nome"));  
        // Recupera a idade do banco e atribui ele ao objeto  
        contato.setIdade(rset.getInt("idade"));  
        // Recupera a data do banco e atribui ela ao objeto  
        contato.setDataCadastro(rset.getDate("dataCadastro"));  
        // Adiciono o contato recuperado, a lista de contatos  
        contatos.add(contato);  
    }  
}
```

## Vamos criar nossa Classe ContatoDAO Parte-10



```

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (rset != null) {
                rset.close();
            }
            if (pstm != null) {
                pstm.close();
            }
            if (conn != null) {
                conn.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return contatos;
}
}

```

Vamos criar nossa Classe ContatoDAO Parte-11



```
import agenda.ContatoDAO;
import java.util.Date;
import agenda.Contato;
public class TesteAgenda {
    public static void main(String[] args) {
        ContatoDAO contatoDAO = new ContatoDAO();
        //Cria um contato e salva no banco
        Contato contato = new Contato();
        contato.setNome("Etevaldo");
        contato.setIdade(56);
        contato.setDataCadastro(new Date());
        contatoDAO.save(contato);
        //Atualiza o contato com id = 1 com os dados do objeto contato1
        Contato contato1 = new Contato();
        contato1.setId(2);
        contato1.setNome("Outro contato");
        contato1.setIdade(42);
        contato1.setDataCadastro(new Date());
    }
}
```

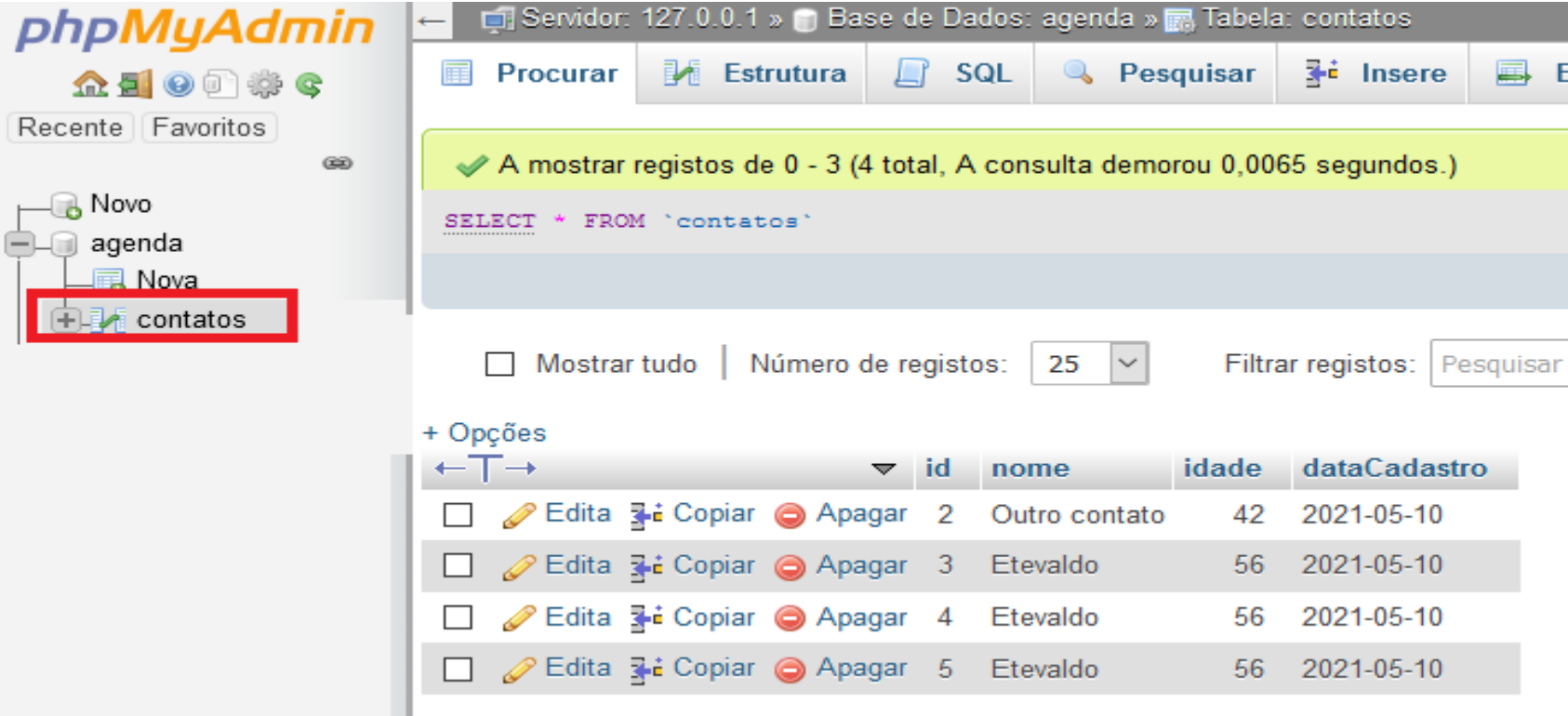
Vamos criar nossa Classe TesteAgenda Parte-01



```
contatoDAO.update(contato1);  
//Remove o contato com id = 1  
contatoDAO.removeById(1);  
//Lista todos os contatos do banco de dados  
for(Contato c : contatoDAO.getContatos()){  
    System.out.println("NOME: " + c.getNome());  
}  
}
```

Vamos criar nossa Classe TesteAgenda Parte-02

Agora só rodar nossa classe TesteAgenda e ver o resultado final.



phpMyAdmin

Recente Favoritos

Novo agenda Nova contatos

Servidor: 127.0.0.1 » Base de Dados: agenda » Tabela: contatos

Procurar Estrutura SQL Pesquisar Inserir

✓ A mostrar registros de 0 - 3 (4 total, A consulta demorou 0,0065 segundos.)

```
SELECT * FROM `contatos`
```

☐ Mostrar tudo | Número de registros: 25 | Filtrar registros: Pesquisar

+ Opções

				id	nome	idade	dataCadastro			
<input type="checkbox"/>	✎	Edita	✚	Copiar	✖	Apagar	2	Outro contato	42	2021-05-10
<input type="checkbox"/>	✎	Edita	✚	Copiar	✖	Apagar	3	Etevaldo	56	2021-05-10
<input type="checkbox"/>	✎	Edita	✚	Copiar	✖	Apagar	4	Etevaldo	56	2021-05-10
<input type="checkbox"/>	✎	Edita	✚	Copiar	✖	Apagar	5	Etevaldo	56	2021-05-10

Agora pegue esta estrutura e faça alteração de todo projeto colocando outras classes modelo para poder inserir no banco de dados.

## Bibliografia base:

<http://brunorota.com.br/2012/05/14/tutorial-criar-crud-em-java-com-jdbc-parte-1/>

<http://brunorota.com.br/2012/05/19/tutorial-criar-crud-em-java-com-jdbc-parte-1-final/>