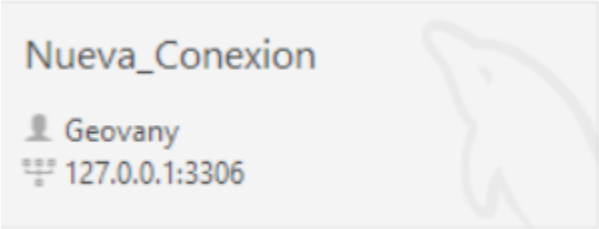
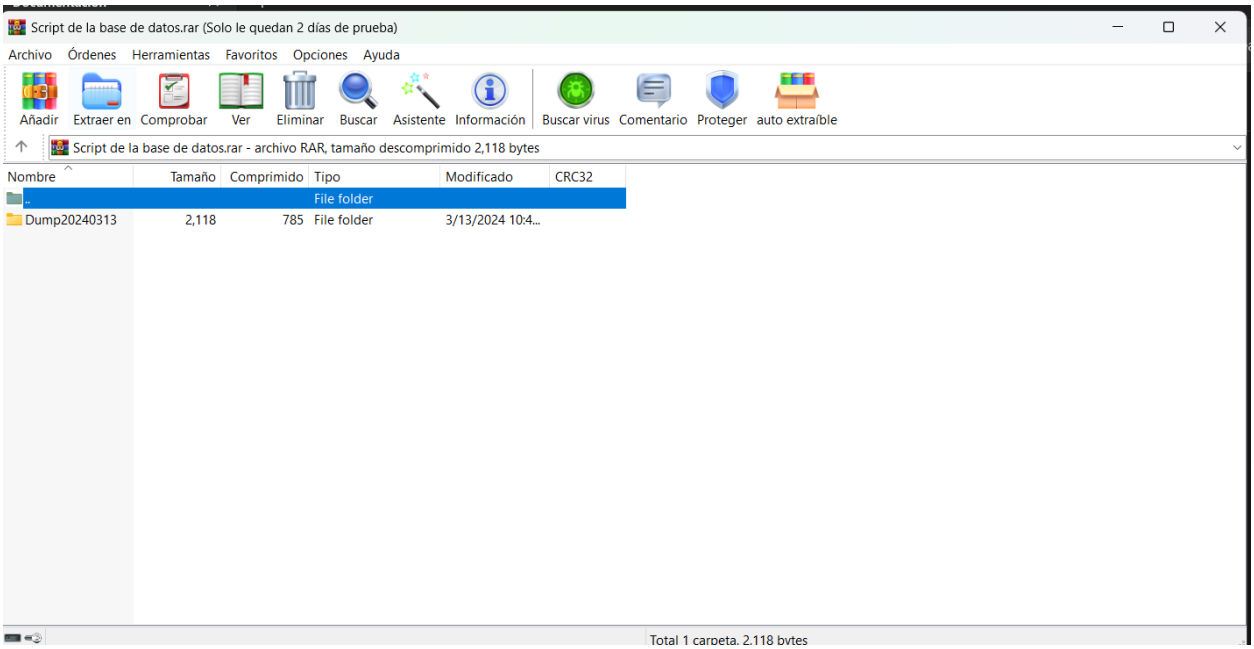
¿Cómo realizo mi conexión a la base de datos?

1. Creamos un nuevo usuario para no trabajar en root

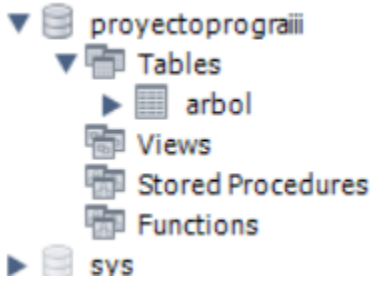


   Link de video guía:
   https://www.youtube.com/watch?v=SB36NEQMzVk

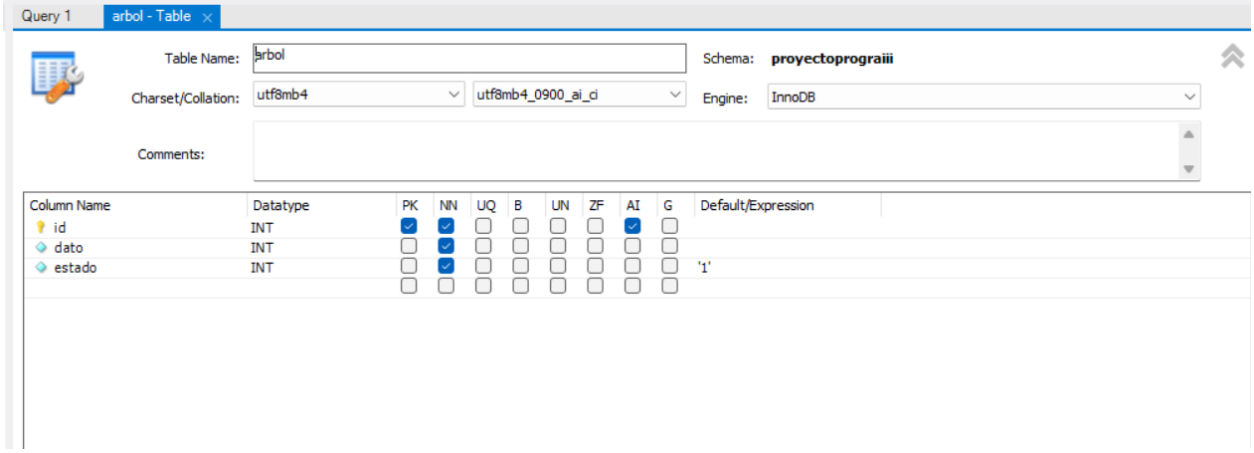
2. Extraemos el script y lo importamos a workbench.



3. Ya importado nos quedará así

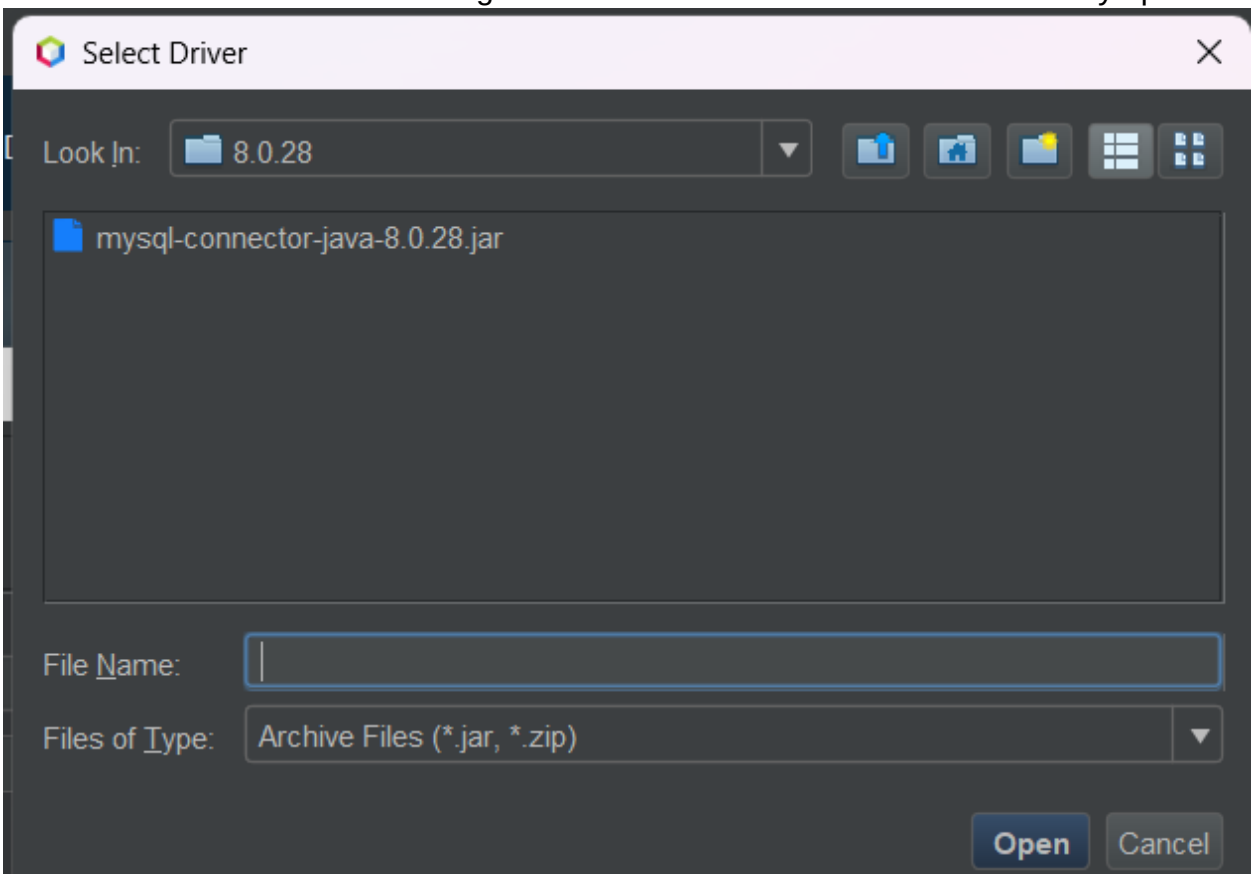


4. Nuestra única tabla es árbol



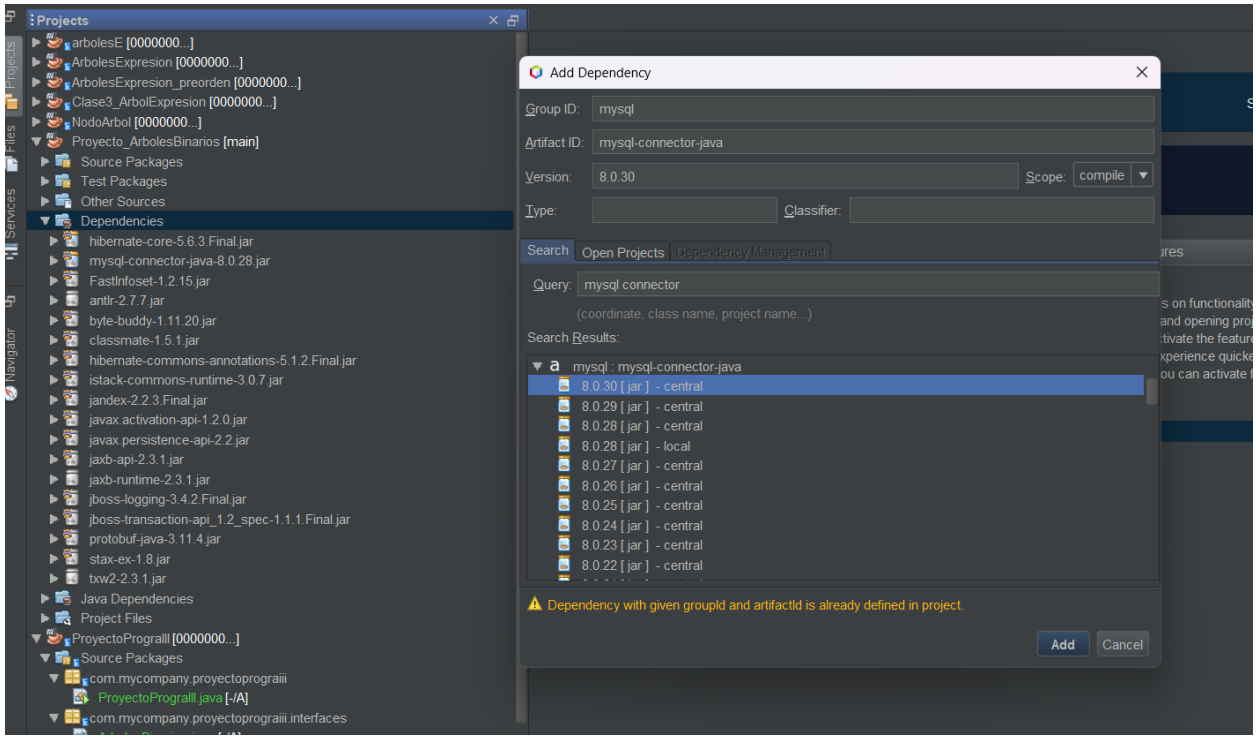
id: Llave primaria que guardará nuestros registros de forma única, es autoincrementadle.
dato: Es un entero, el cual se insertará por cada uno que se lea del archivo de texto.
estado: Este representa el estado del registro, 1=activo, 0=inactivo, lo cual es buena práctica para no perder registros y auditorias de sistemas.

5. Debemos de descargar el driver de mysql

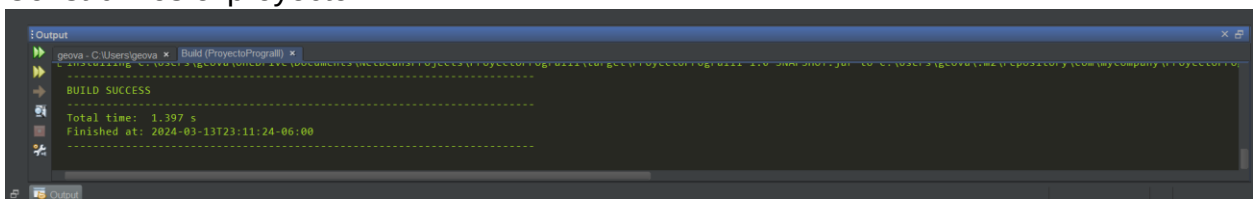


6. Database – new connection

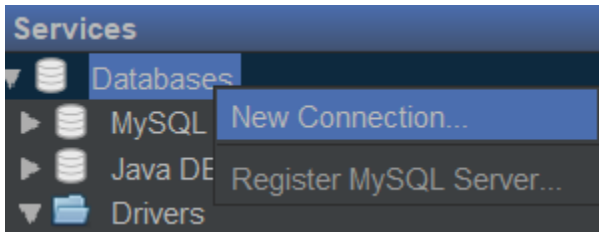
7. Seleccionamos el driver



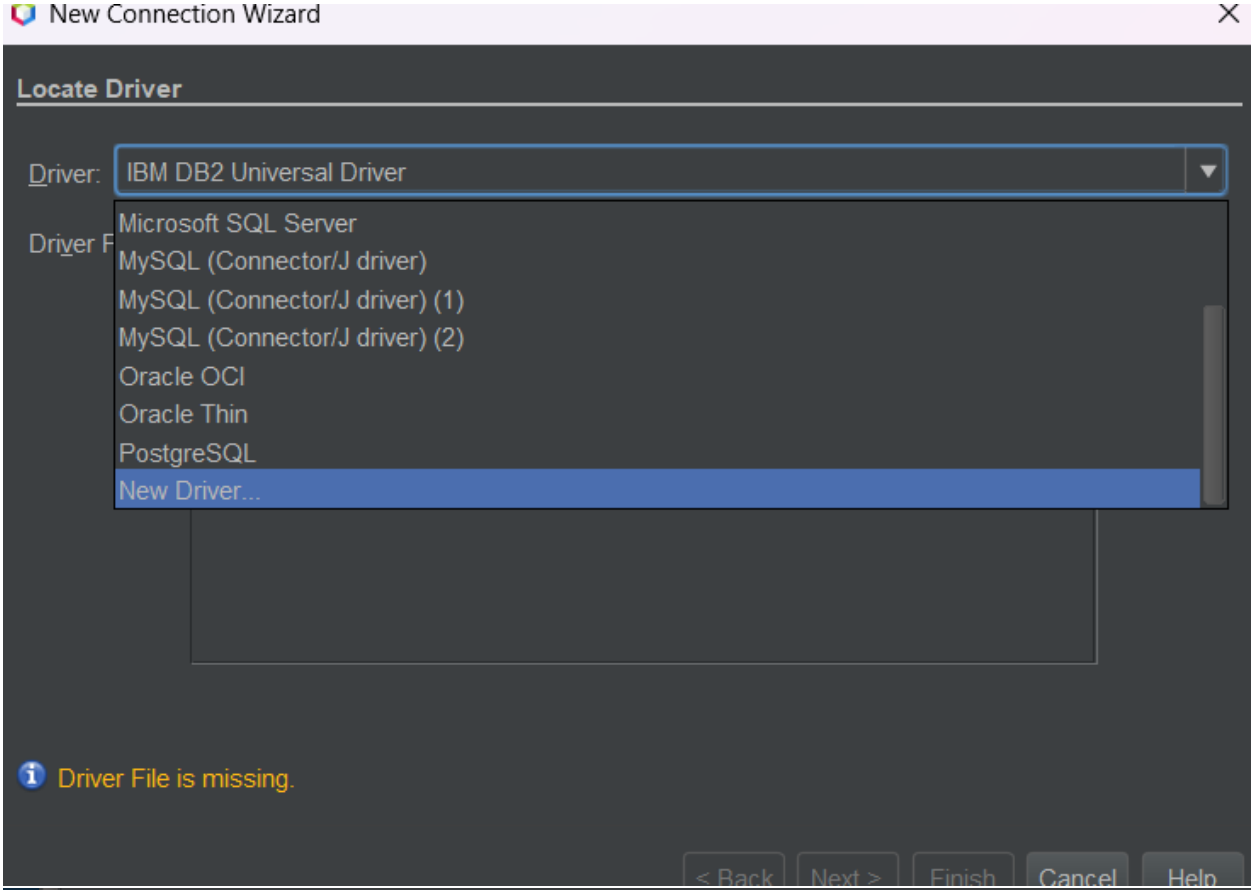
8. Construimos el proyecto

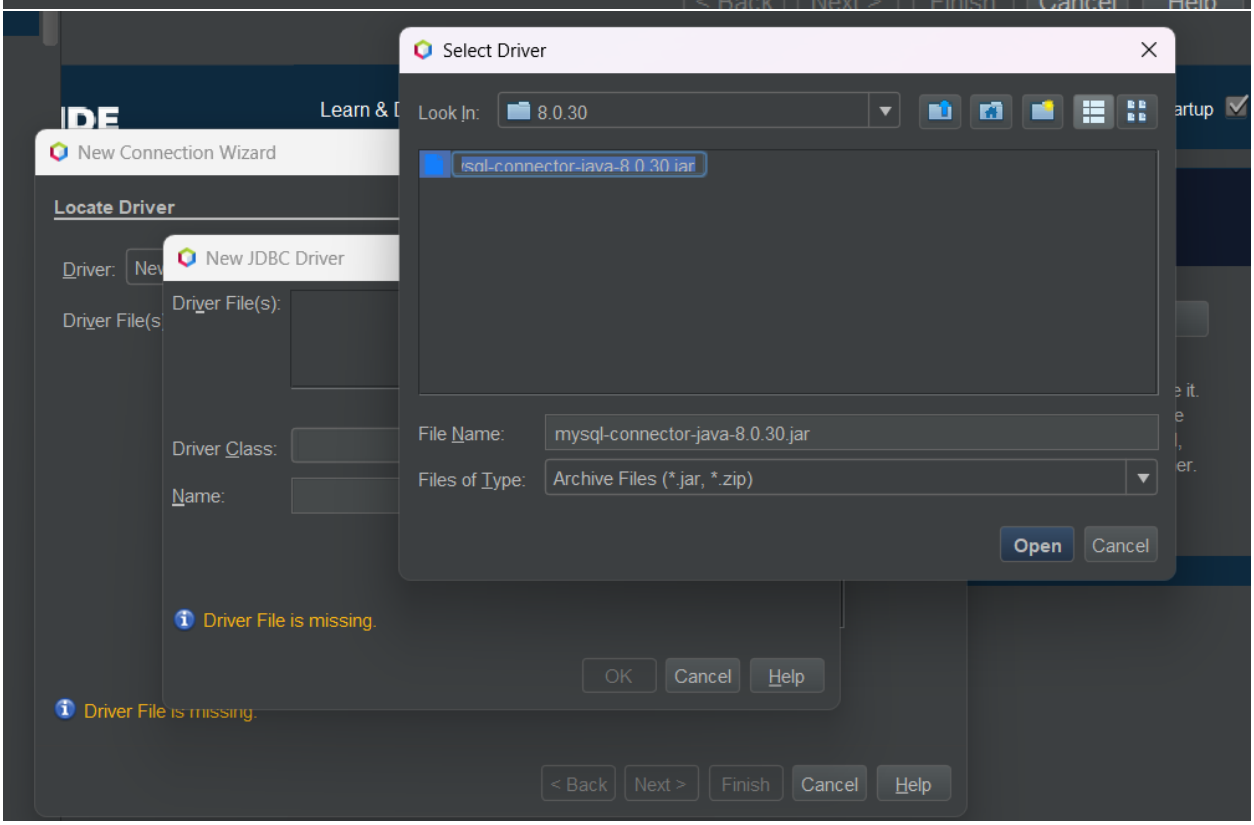


9. Ahora nos vamos a services – Databases – New Connection

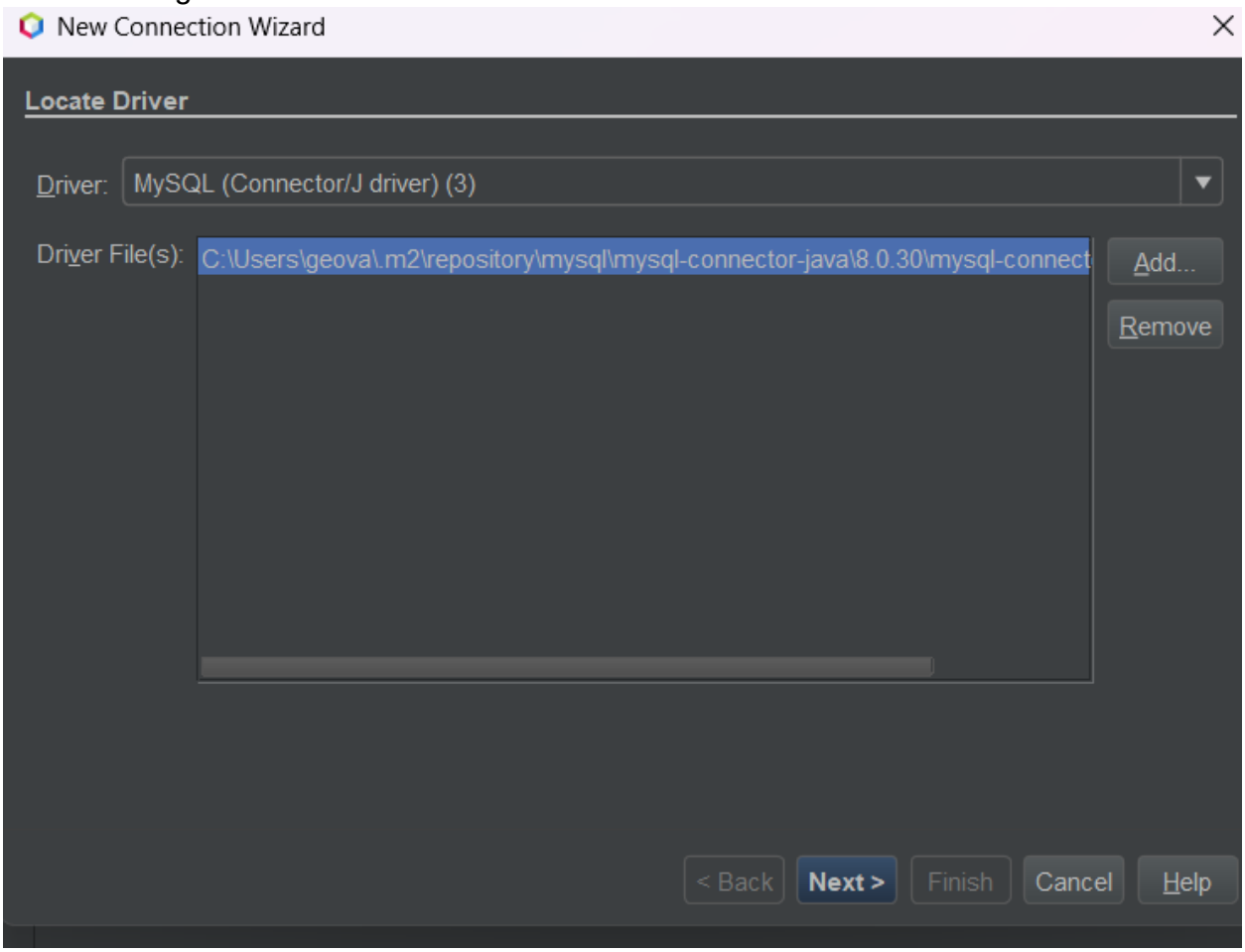


10. Agregamos el driver que acabamos de descargar

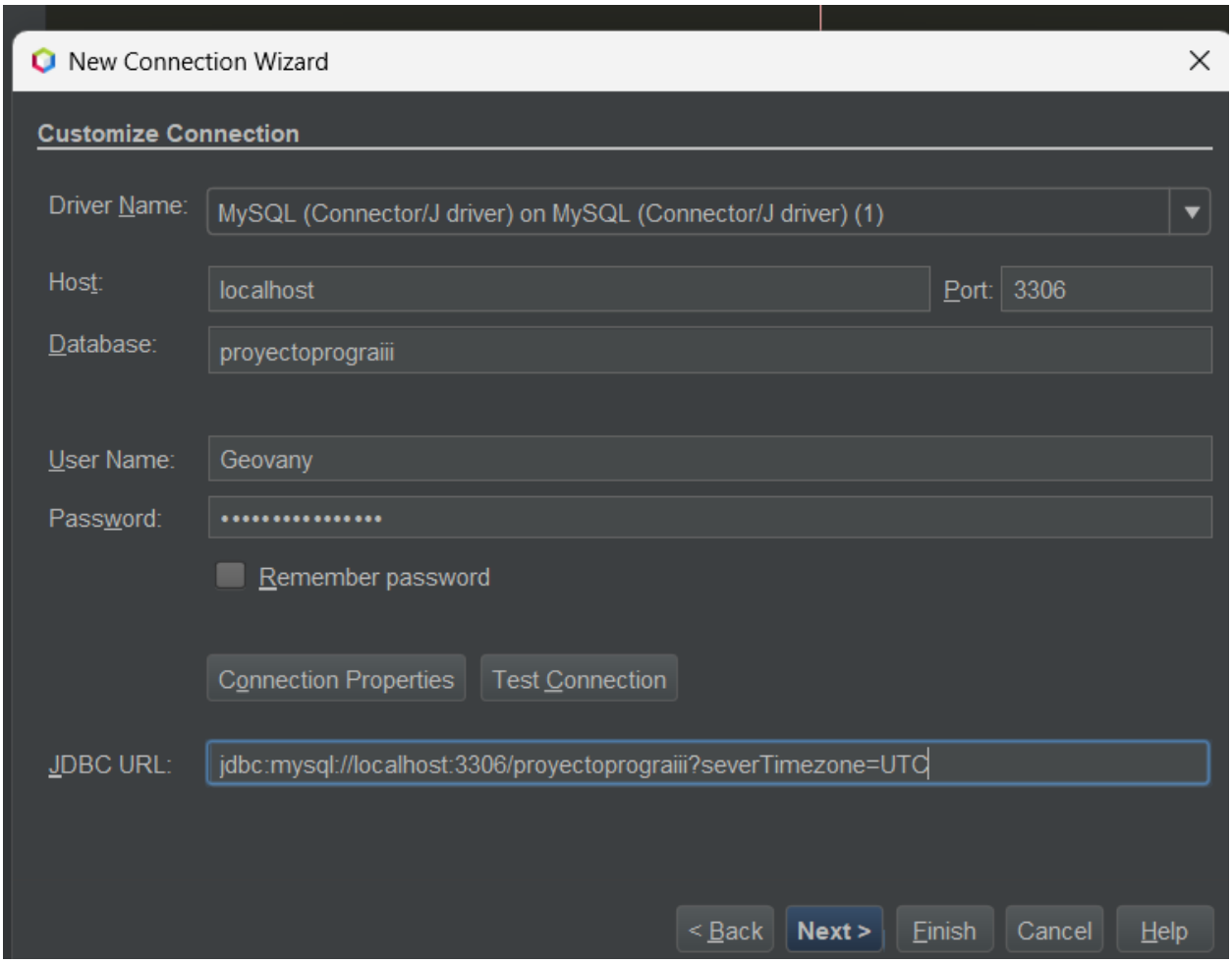

New Connection Wizard
Locate Driver
Driver:
IBM DB2 Universal Driver
Microsoft SQL Server
MySQL (Connector/J driver)
MySQL (Connector/J driver) (1)
MySQL (Connector/J driver) (2)
Oracle OCI
Oracle Thin
PostgreSQL
New Driver...
Driver File is missing.
< Back
Next >
Finish
Cancel
Help


Select Driver
Look In:
8.0.30
mysql-connector-java-8.0.30.jar
File Name:
mysql-connector-java-8.0.30.jar
Files of Type:
Archive Files (*.jar, *.zip)
Open
Cancel
New Connection Wizard
Locate Driver
Driver:
New JDBC Driver
Driver File(s):
Driver Class:
Name:
Driver File is missing.
OK
Cancel
Help
Driver File is missing.
< Back
Next >
Finish
Cancel
Help

11. Le damos siguiente

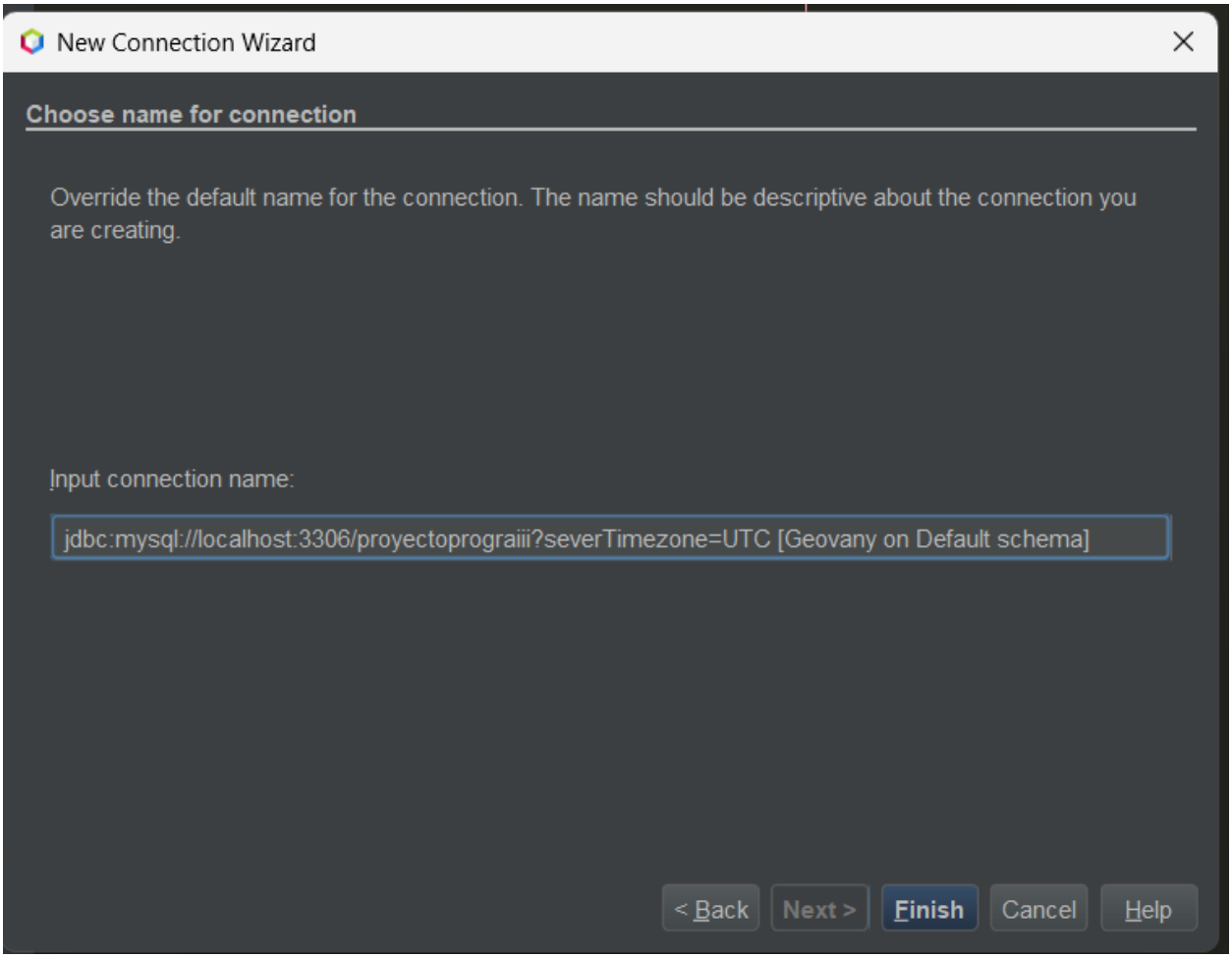


12. Llenamos datos

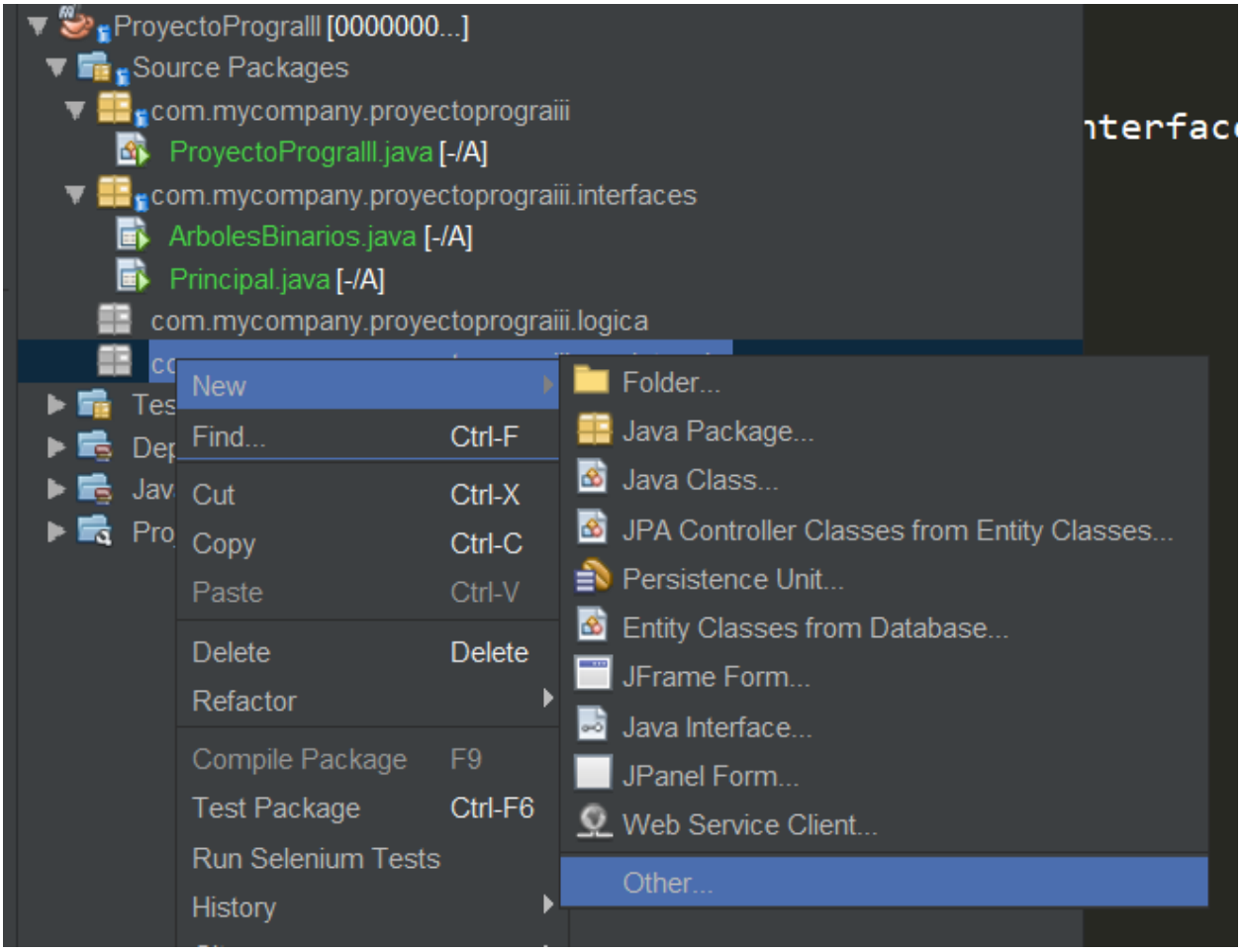


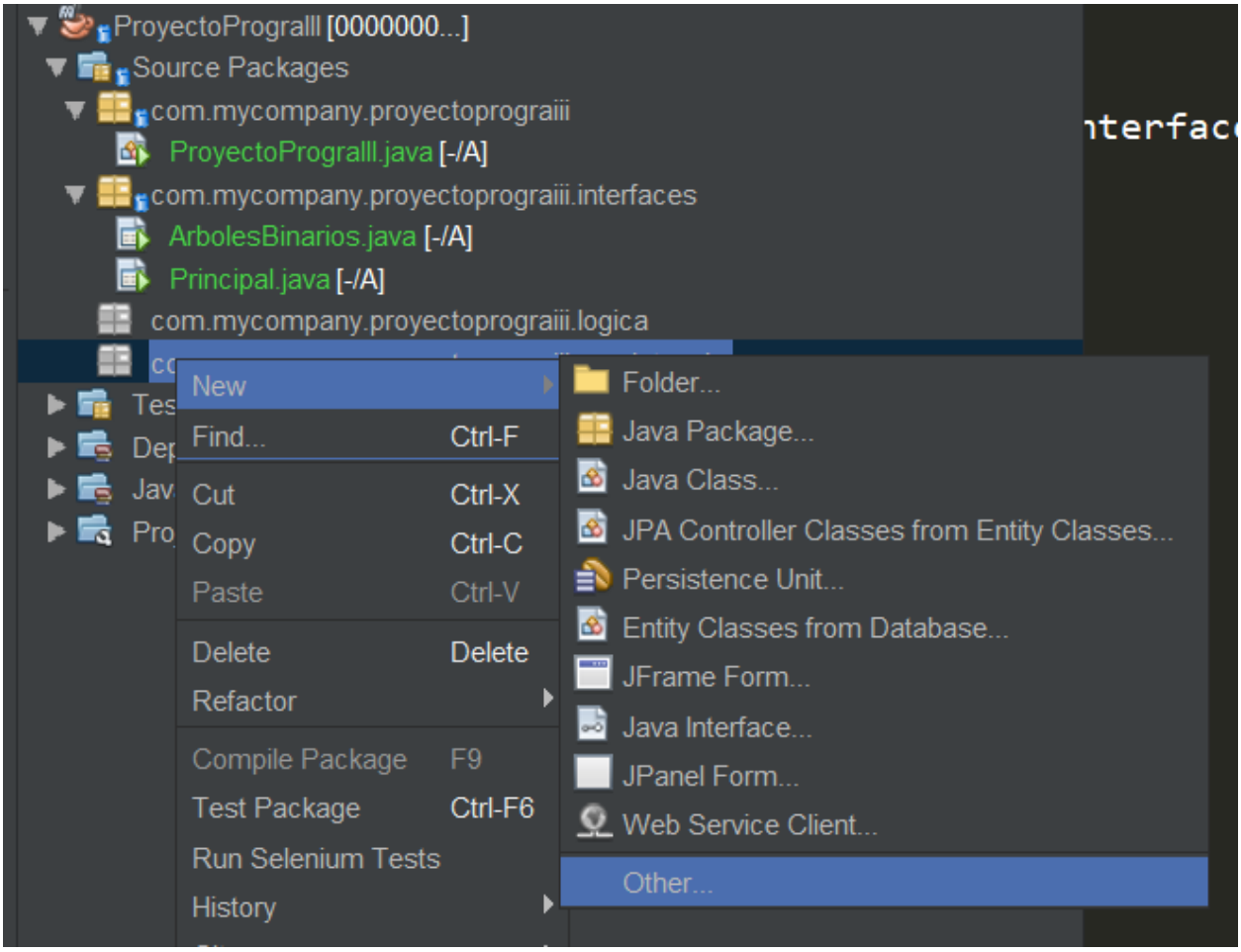
El serverTimeZone nos sirve para que no hayan errores con los horarios.

13. Avanzamos y le damos finish

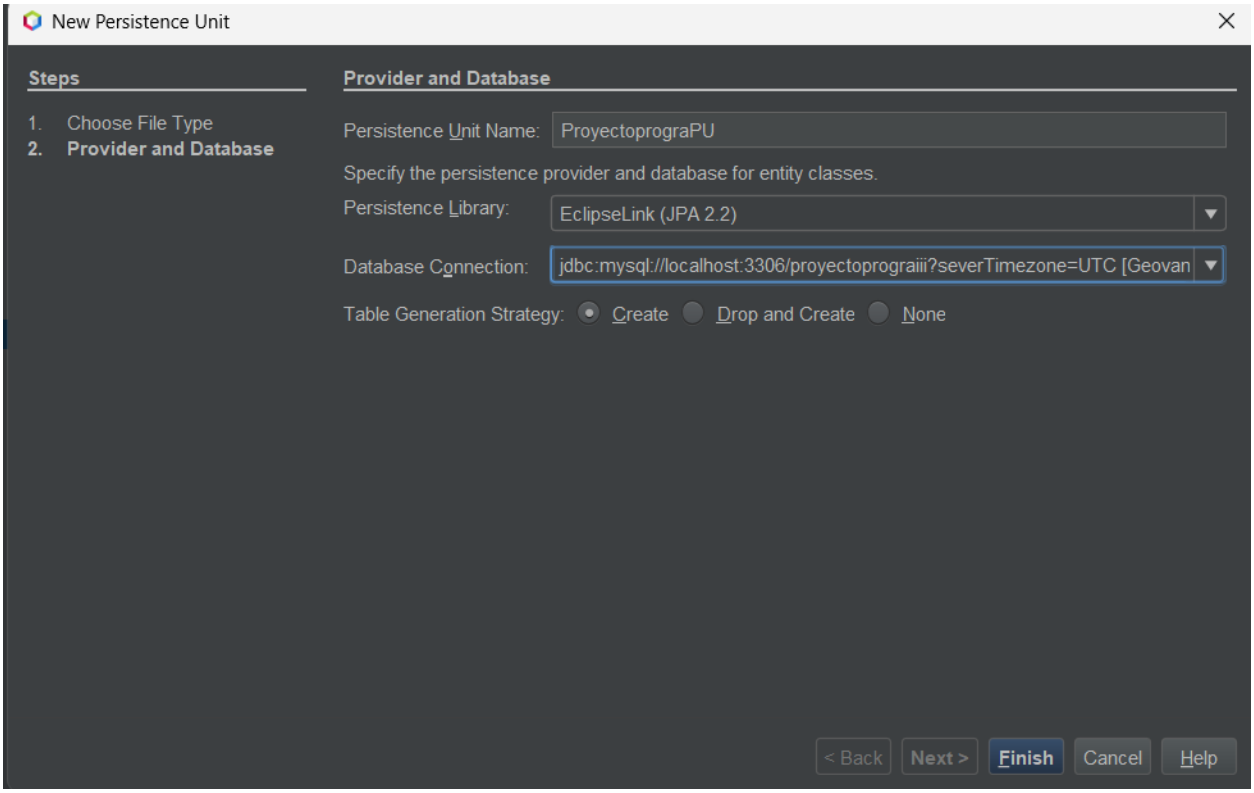


14. Ahora crearemos el archivo de persistence

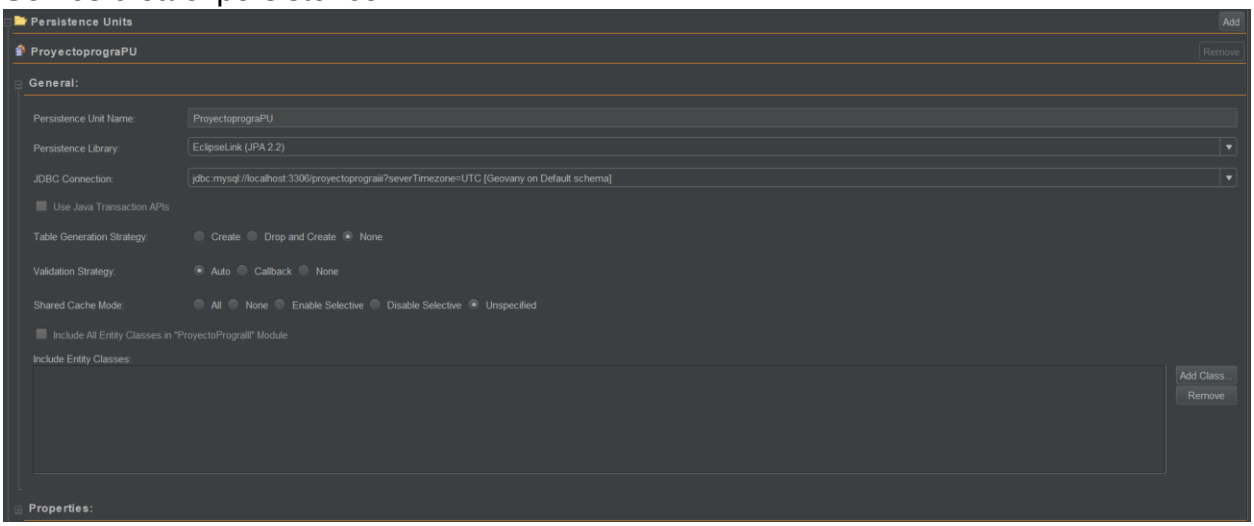

ProyectoProgralll [0000000...]
Source Packages
com.mycompany.proyectoprograiii
ProyectoProgralll.java [-/A]
com.mycompany.proyectoprograiii.interfaces
ArbolesBinarios.java [-/A]
Principal.java [-/A]
com.mycompany.proyectoprograiii.logica
New
Find... Ctrl-F
Cut Ctrl-X
Copy Ctrl-C
Paste Ctrl-V
Delete Delete
Refactor
Compile Package F9
Test Package Ctrl-F6
Run Selenium Tests
History
Folder...
Java Package...
Java Class...
JPA Controller Classes from Entity Classes...
Persistence Unit...
Entity Classes from Database...
JFrame Form...
Java Interface...
JPanel Form...
Web Service Client...
Other...



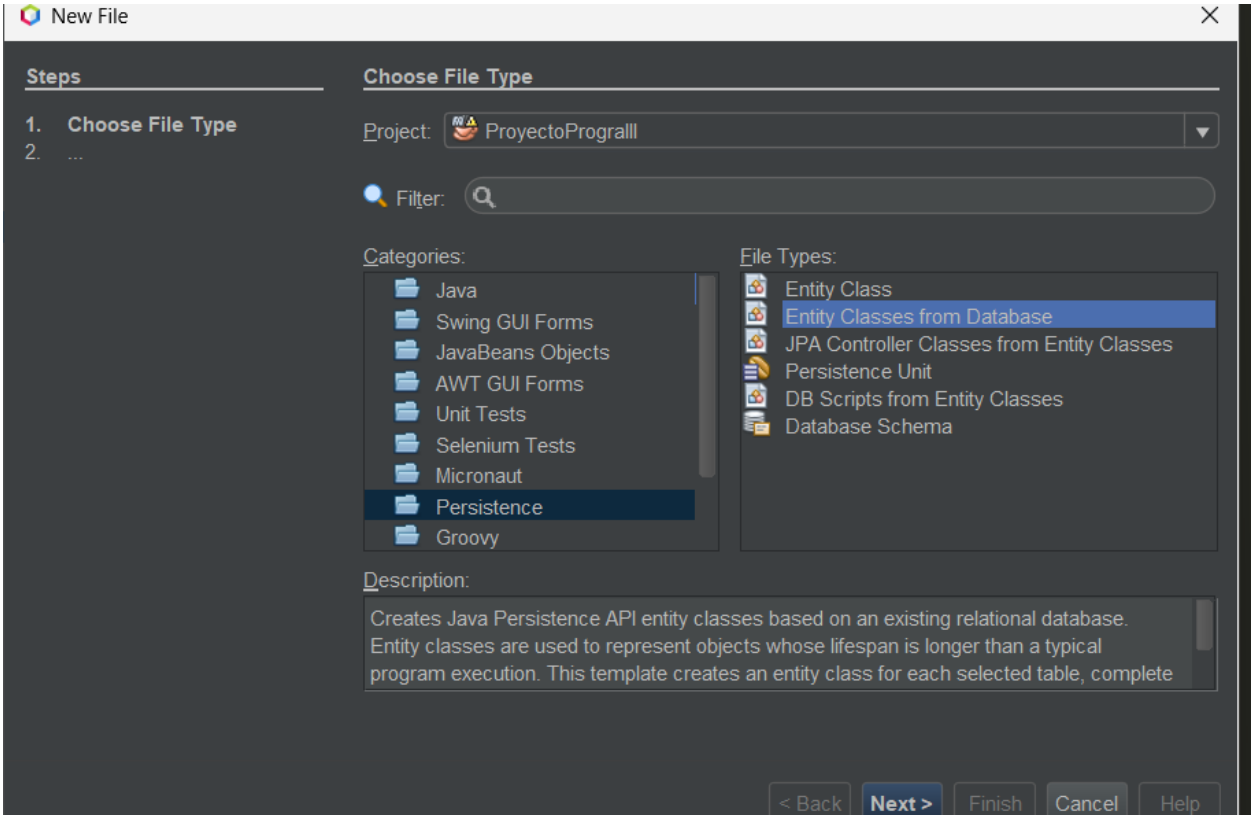
15. Llenamos los datos



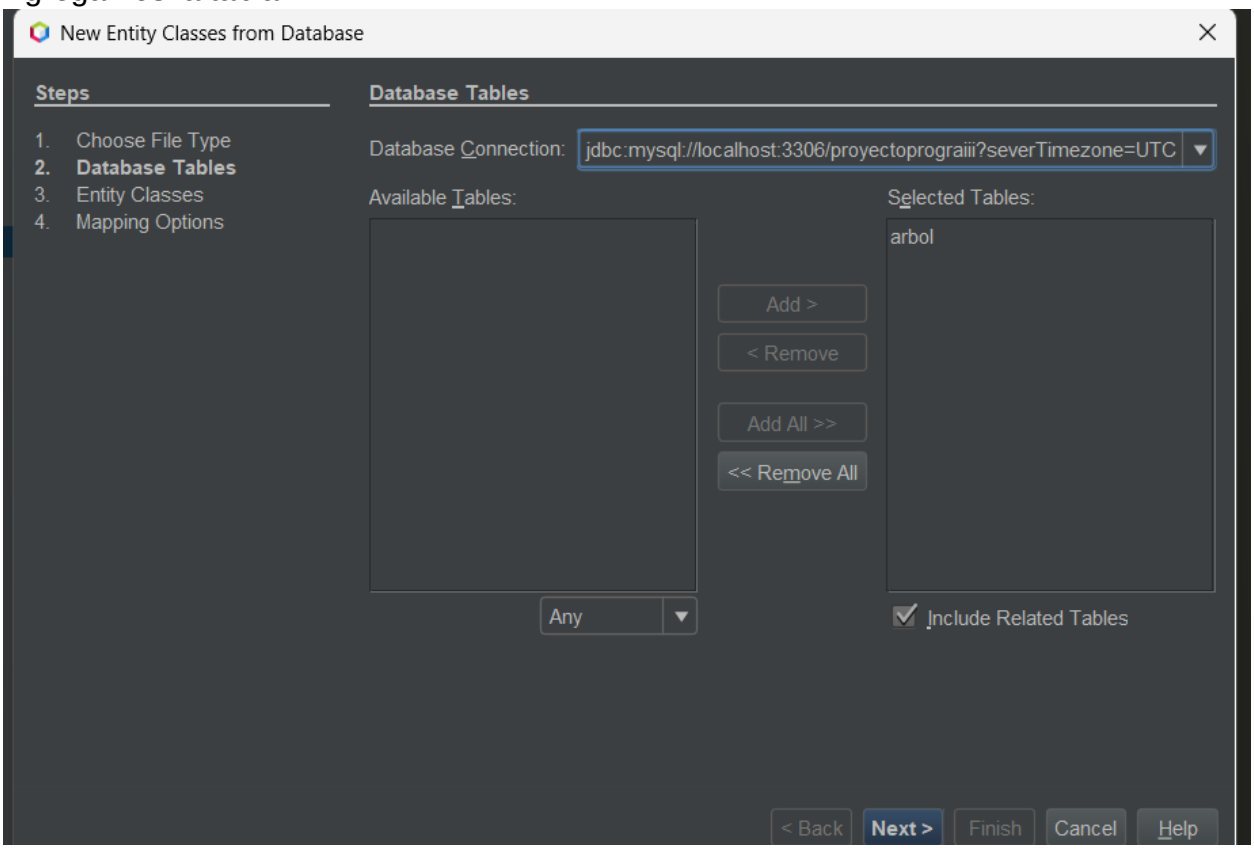
16. Se nos crea el persistence



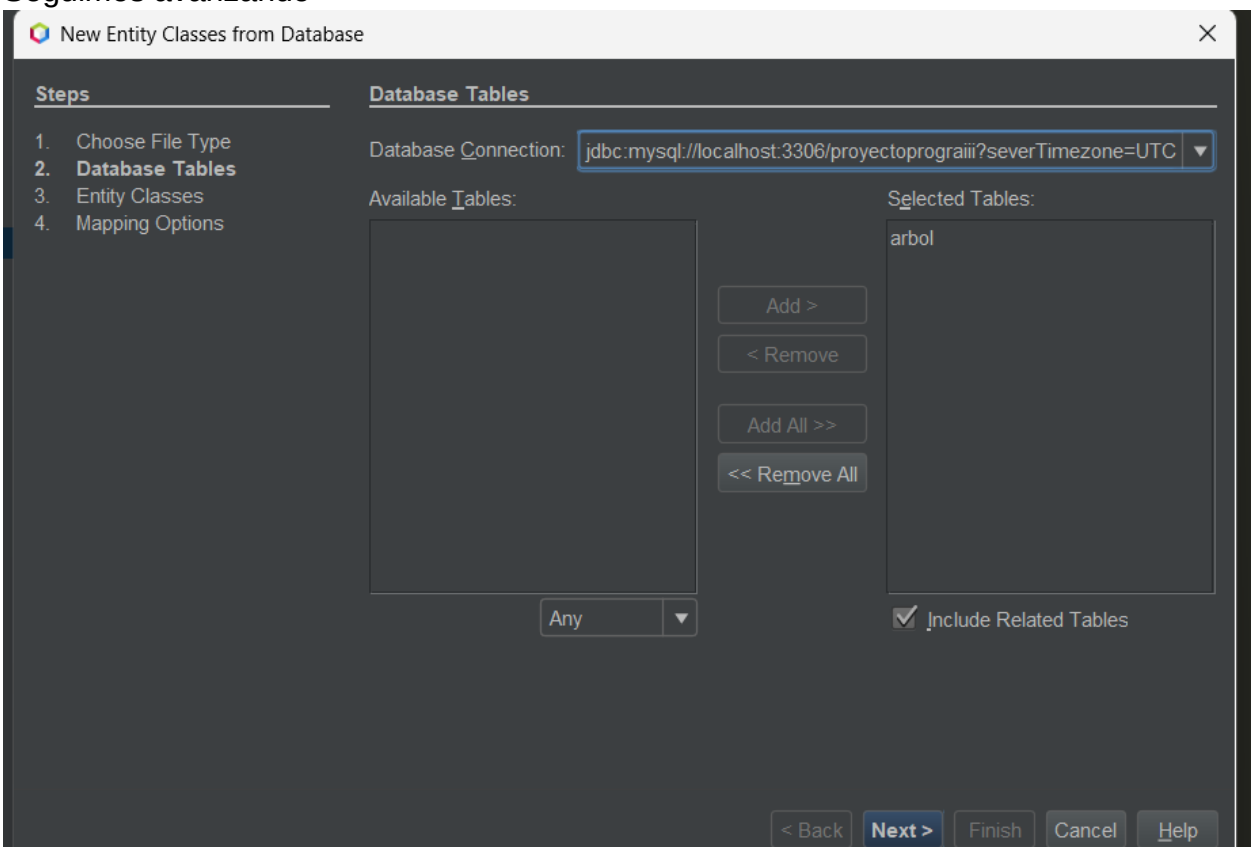
17. Crearemos la entity classes



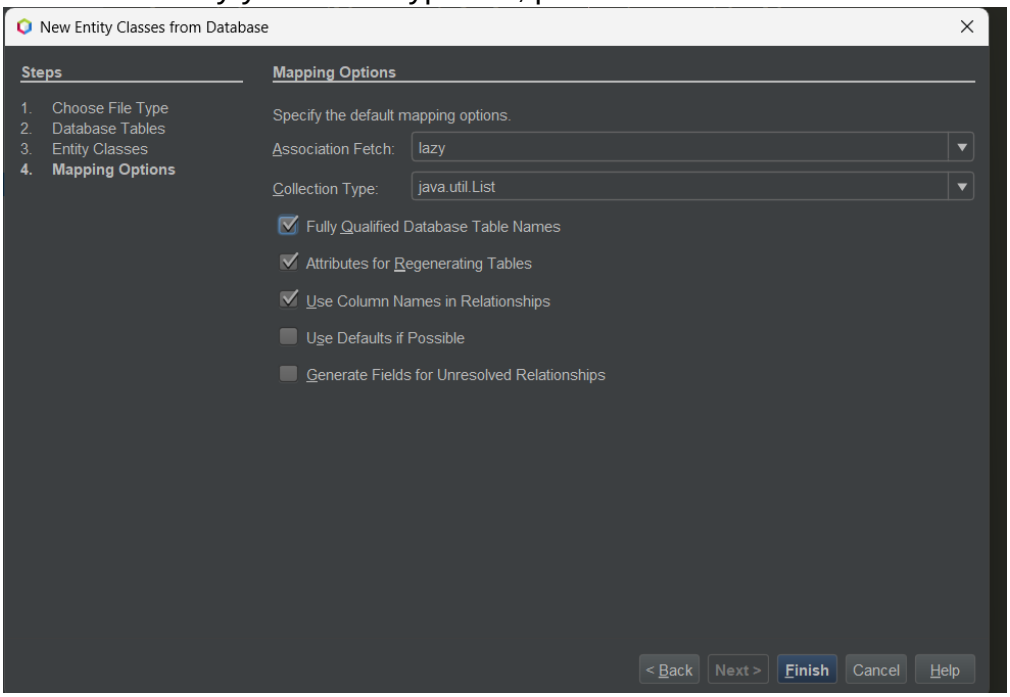
18. Agregamos la tabla.



19. Seguimos avanzando



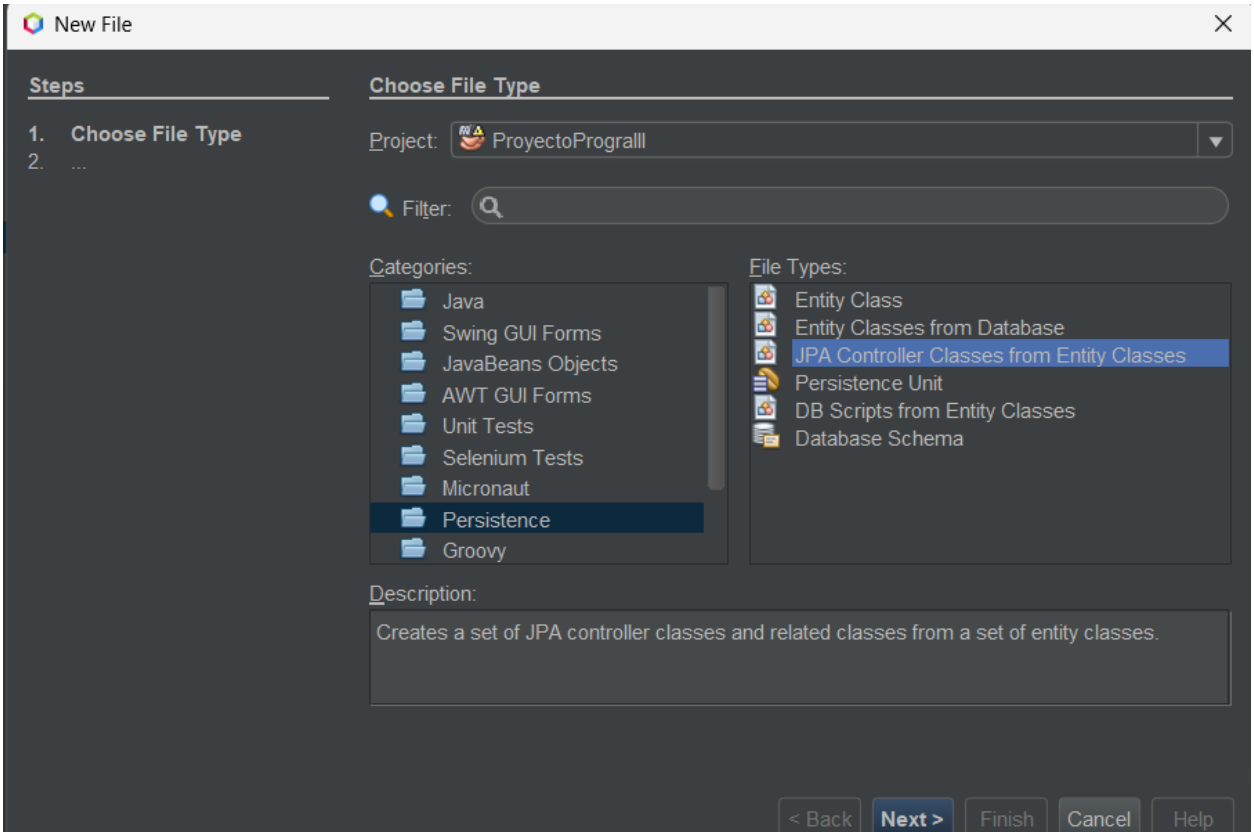
20. Asociación: lazy y Colletion type: list, para usar el método de lista



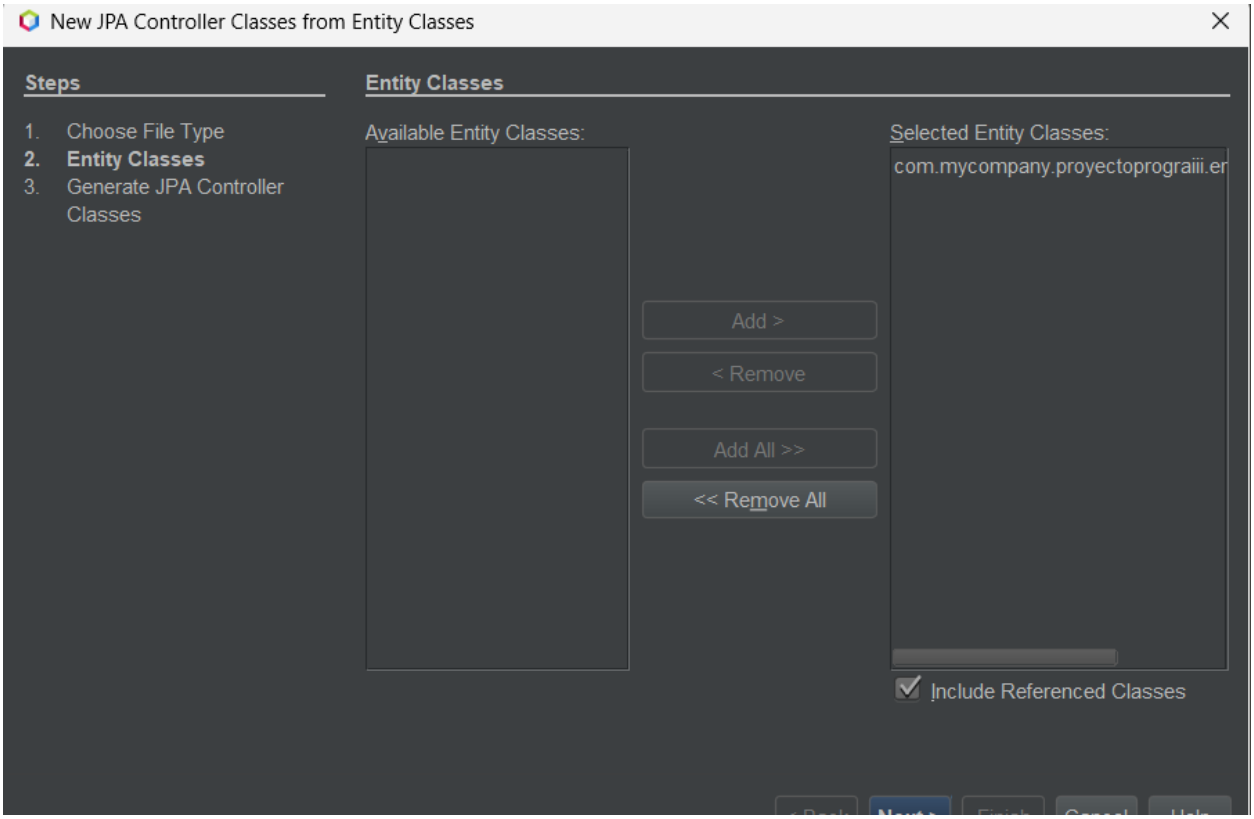
21. Ya la tenemos

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this licen
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.proyectoprograiii.entityclasses;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author geova
 */
@Entity
@Table(name = "arbol", catalog = "proyectoprograiii", schema = "")
@NamedQueries({
    @NamedQuery(name = "Arbol.findAll", query = "SELECT a FROM Arbol a")})
public class Arbol implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
```

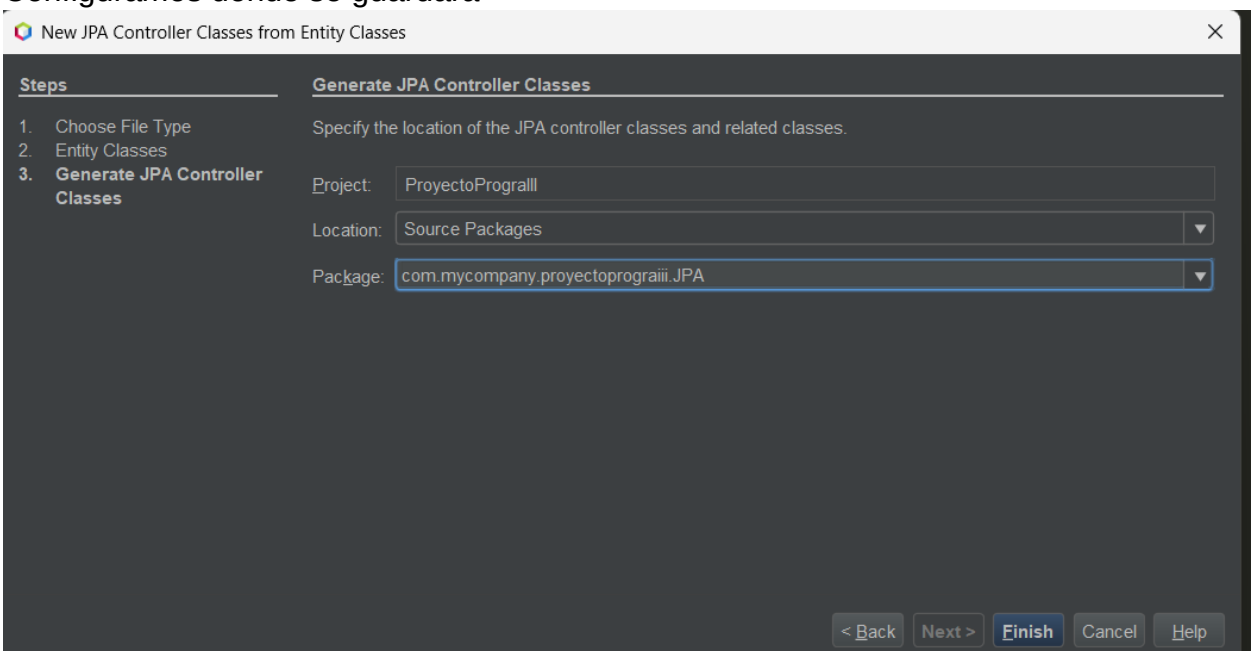
22. Ahora crearemos el jpa



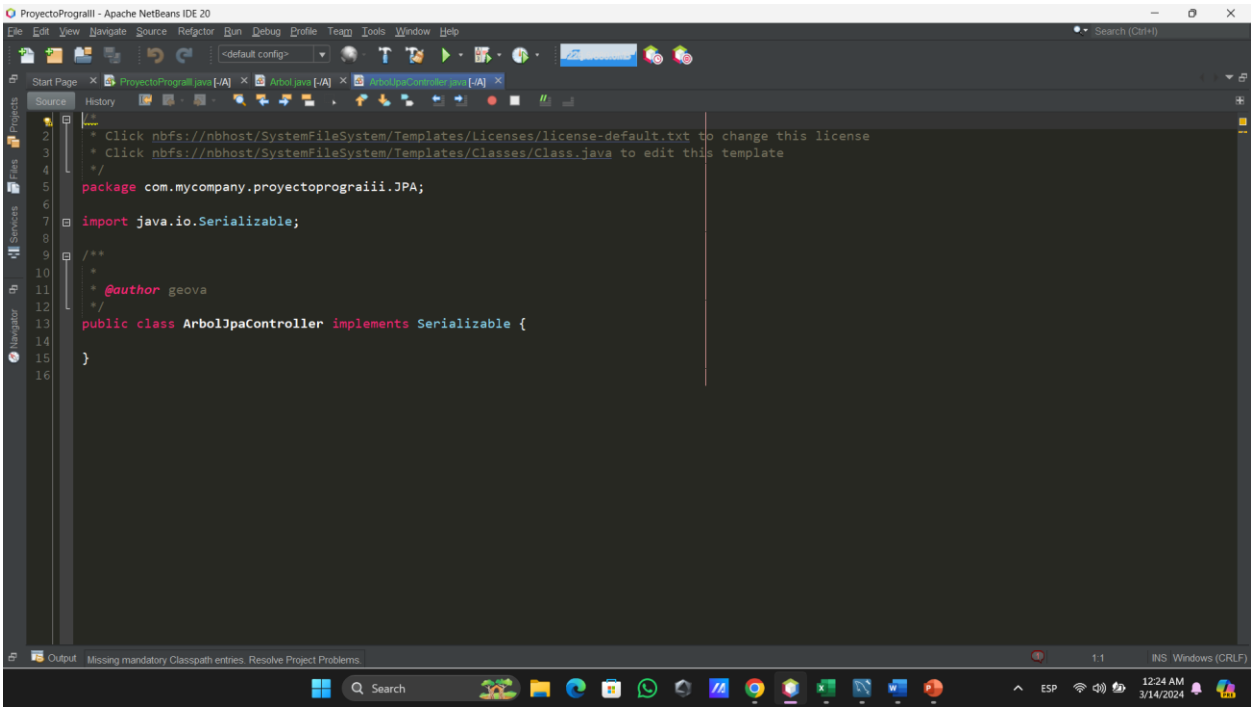
23. Pasamos las clases



24. Configuramos donde se guardará



25. Creado el JPA



26. Pegar este código

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
```

```
package JPA;

import Entityclases.Arbol;
import java.io.Serializable;

/**
 *
 * @author geova
 */
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Query;

public class ArbolJpaController {

    private final EntityManagerFactory entityManagerFactory;

    public ArbolJpaController(EntityManagerFactory entityManagerFactory) {
        this.entityManagerFactory = entityManagerFactory;
    }

    public void create(Arbol arbol) {
        EntityManager                                entityManager                                =
entityManagerFactory.createEntityManager();
        EntityTransaction transaction = entityManager.getTransaction();
        try {
            transaction.begin();
            entityManager.persist(arbol);
            transaction.commit();
        } catch (Exception ex) {
            if (transaction.isActive()) {
                transaction.rollback();
            }
            ex.printStackTrace();
        } finally {
            entityManager.close();
        }
    }

    public Arbol find(Integer id) {
```

```
        EntityManager                                   entityManager                                   =
entityManagerFactory.createEntityManager();
        try {
            return entityManager.find(Arbol.class, id);
        } finally {
            entityManager.close();
        }
    }

    public void update(Arbol arbol) {
        EntityManager                                   entityManager                                   =
entityManagerFactory.createEntityManager();
        EntityTransaction transaction = entityManager.getTransaction();
        try {
            transaction.begin();
            entityManager.merge(arbol);
            transaction.commit();
        } catch (Exception ex) {
            if (transaction.isActive()) {
                transaction.rollback();
            }
            ex.printStackTrace();
        } finally {
            entityManager.close();
        }
    }

    public void delete(Integer id) {
        EntityManager                                   entityManager                                   =
entityManagerFactory.createEntityManager();
        EntityTransaction transaction = entityManager.getTransaction();
        try {
            transaction.begin();
            Arbol arbol = entityManager.find(Arbol.class, id);
            if (arbol != null) {
                entityManager.remove(arbol);
            }
            transaction.commit();
        } catch (Exception ex) {
            if (transaction.isActive()) {
                transaction.rollback();
            }
            ex.printStackTrace();
```

```
        } finally {
            entityManager.close();
        }
    }

    public List<Arbol> findAll() {
        EntityManager entityManager =
entityManagerFactory.createEntityManager();
        try {
            Query query = entityManager.createQuery("SELECT a FROM Arbol a",
Arbol.class);
            return query.getResultList();
        } finally {
            entityManager.close();
        }
    }
}
```