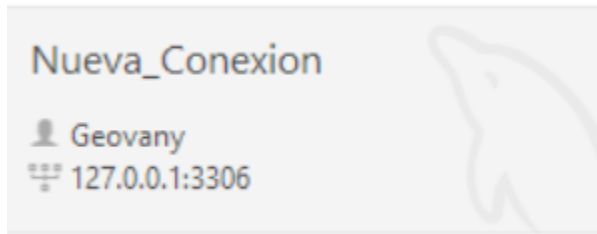


## ¿Cómo realizo mi conexión a la base de datos?

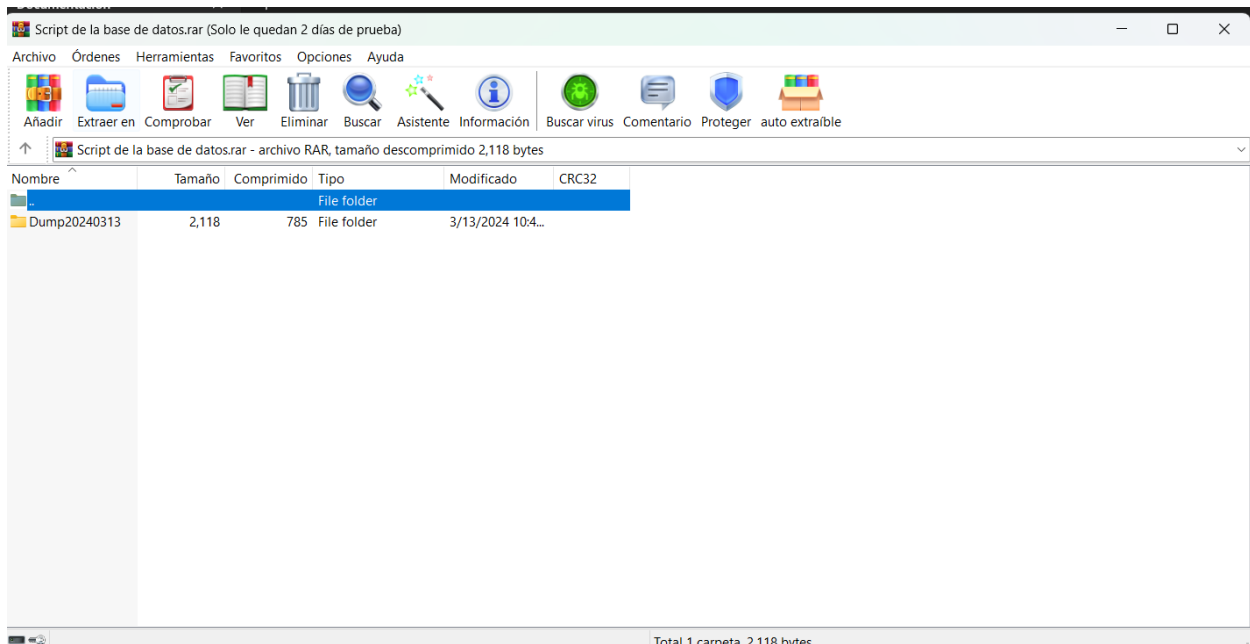
1. Creamos un nuevo usuario para no trabajar en root



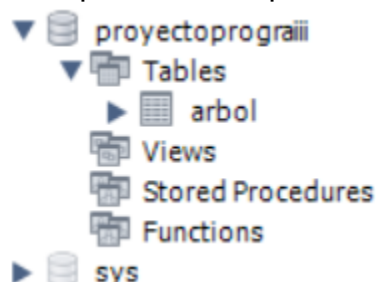
Link de video guía:

<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

Query 1   **arbol - Table**

Table Name:  Schema: **proyectoprograiii**

Charset/Collation:   Engine:

Comments:

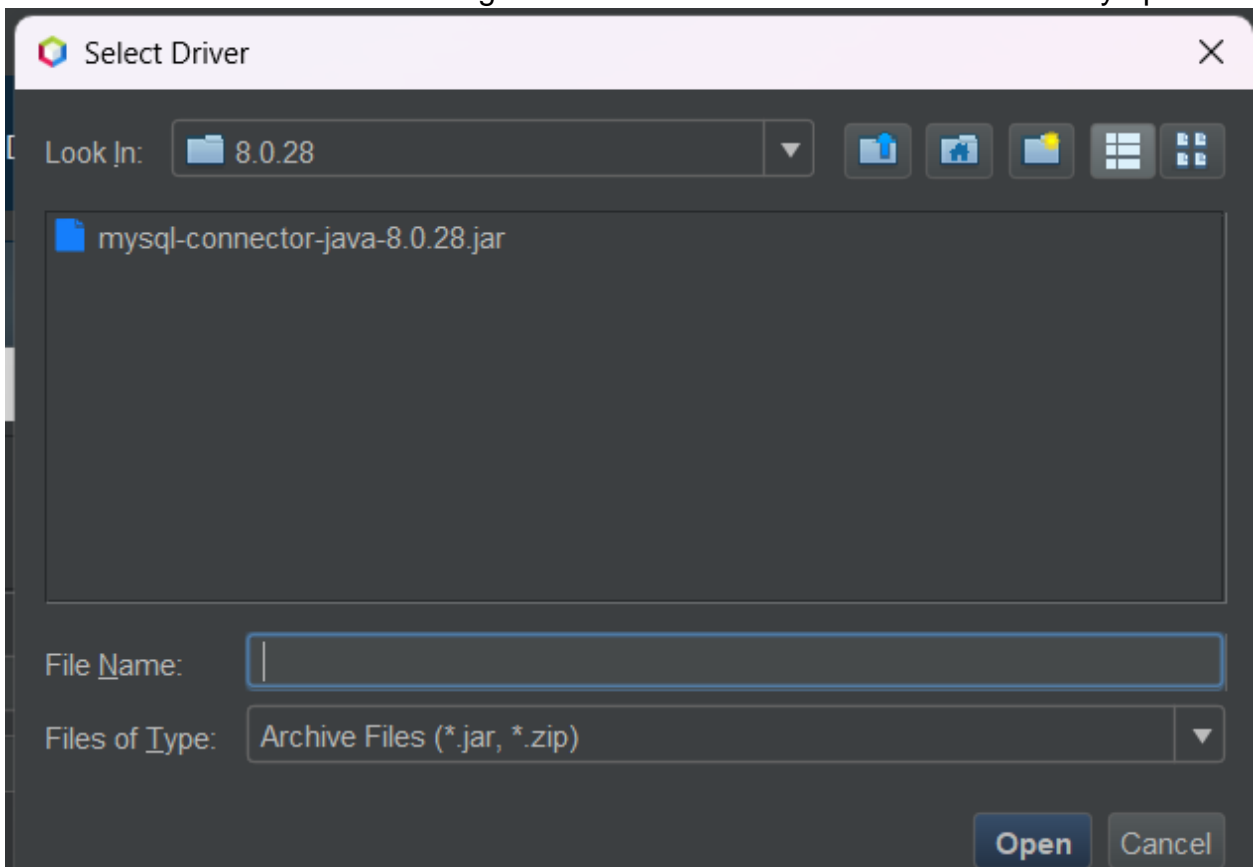
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
dato	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
estado	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	'1'

id: Llave primaria que guardará nuestros registros de forma única, es autoincrementable.

dato: Es un entero, el cual se insertará por cada uno que se lea del archivo de texto.

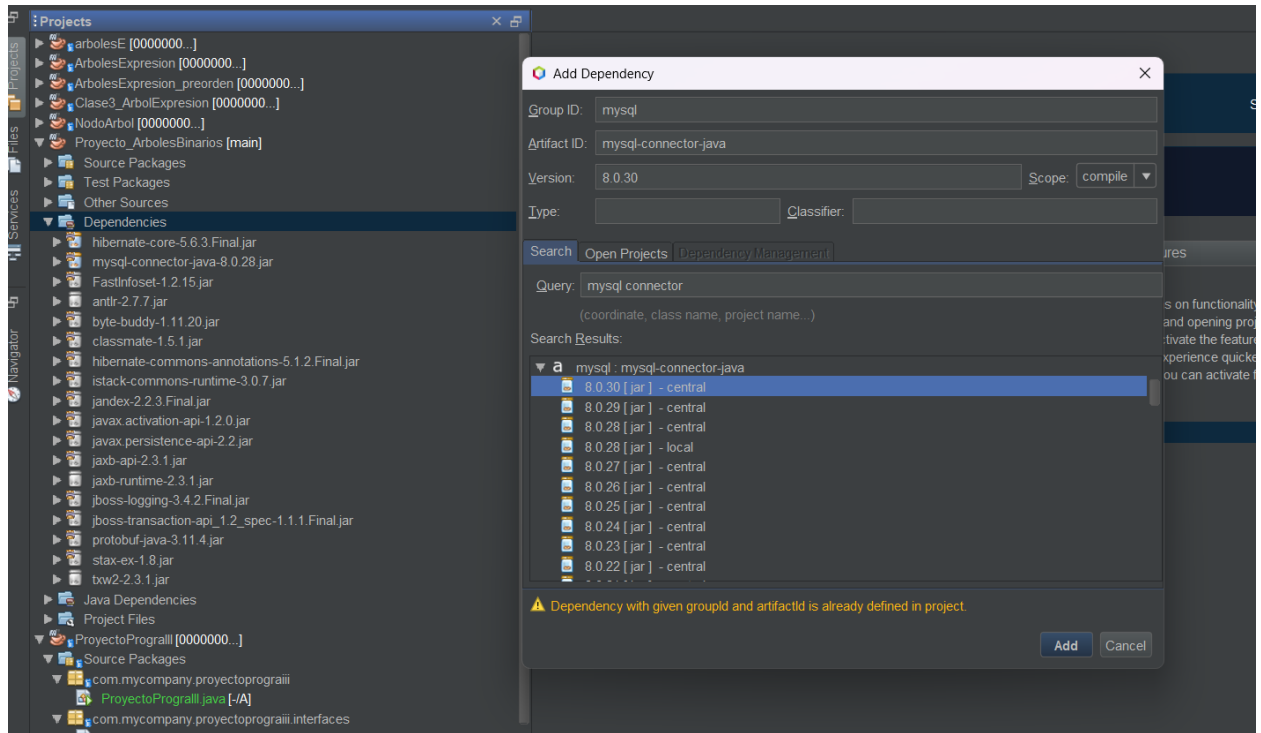
estado: Este representa el estado del registro, 1=activo, 0=inactivo, lo cual es buena práctica para no perder registros y auditorias de sistemas.

5. Debemos de descargar el driver de mysql

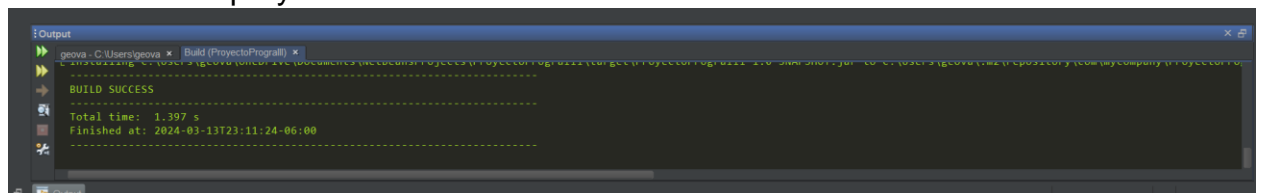


6. Database – new connection

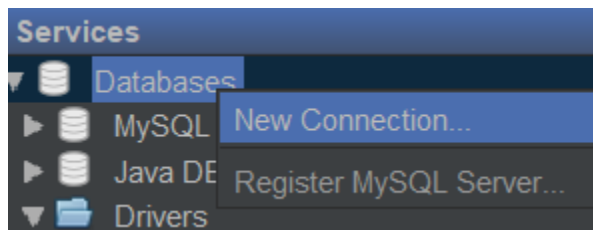
7. Seleccionamos el driver



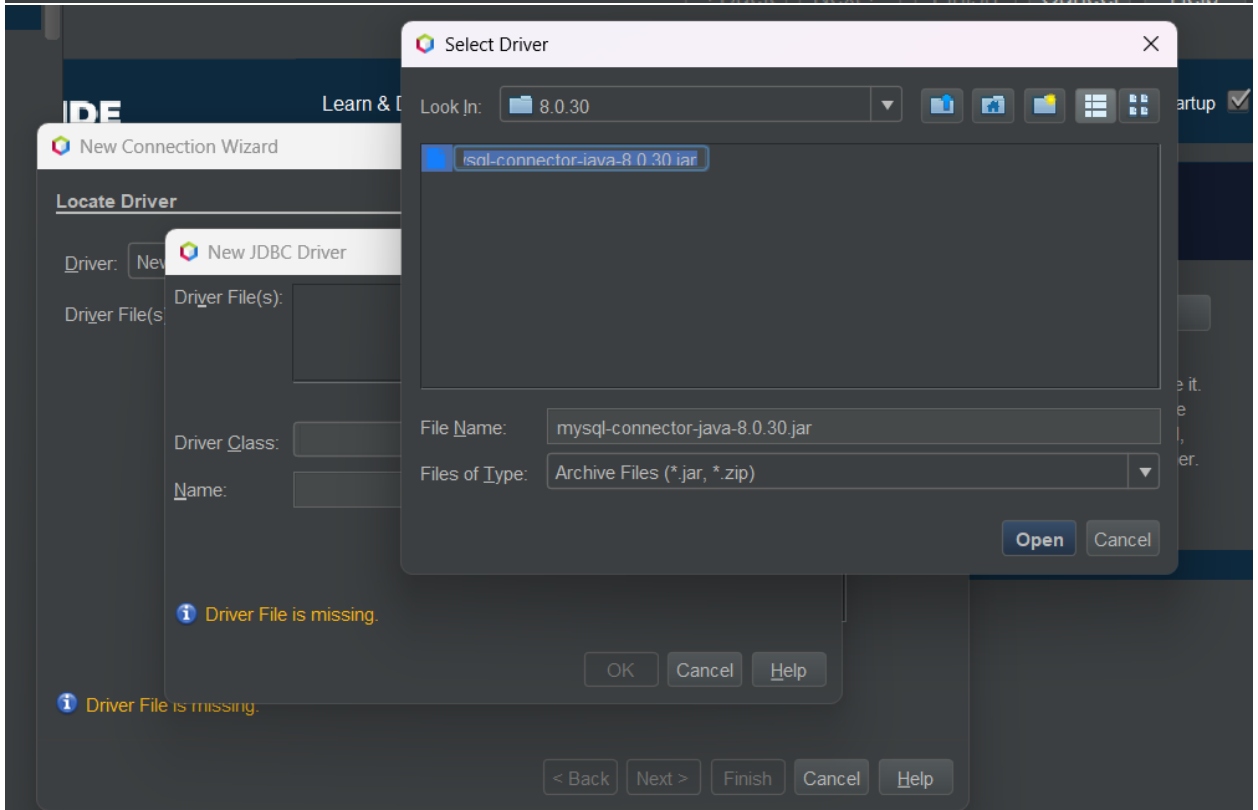
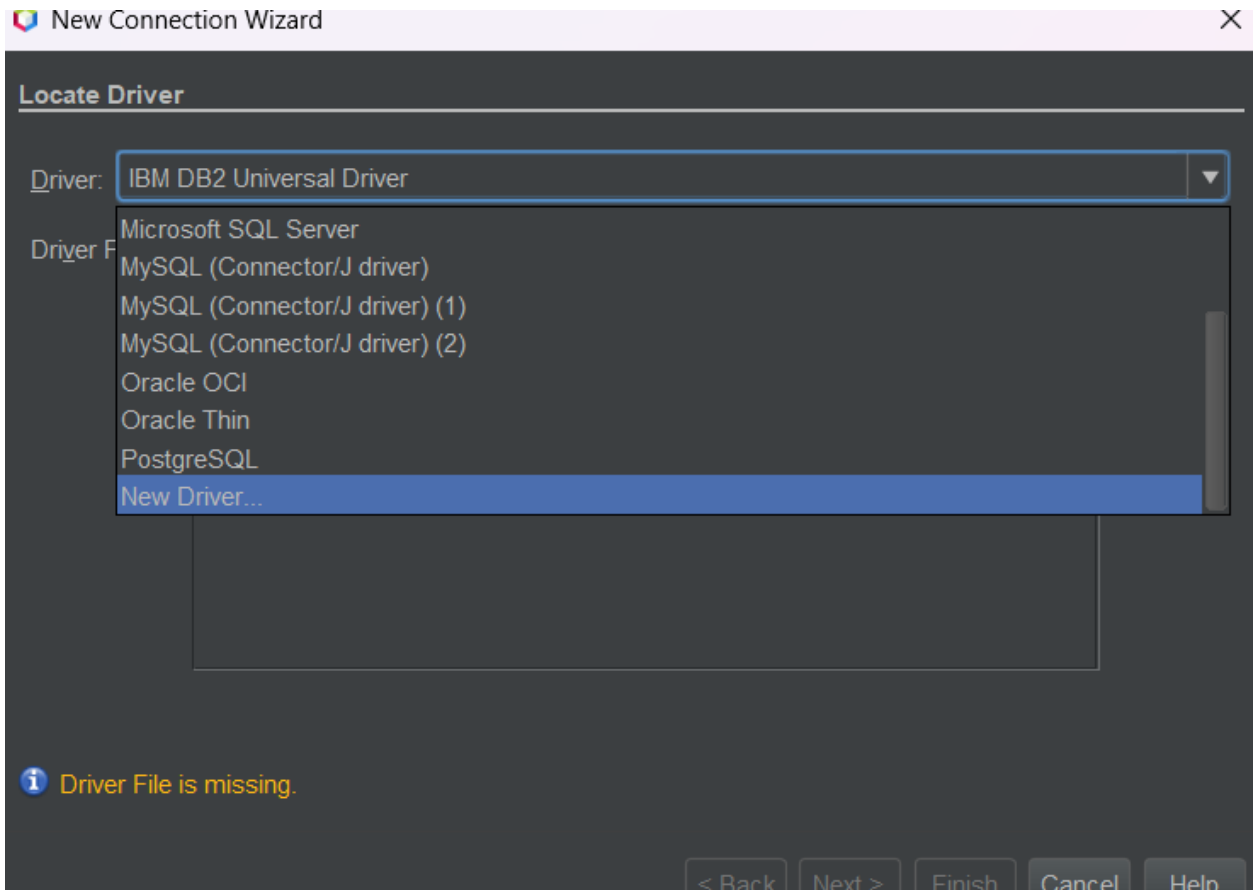
8. Construimos el proyecto



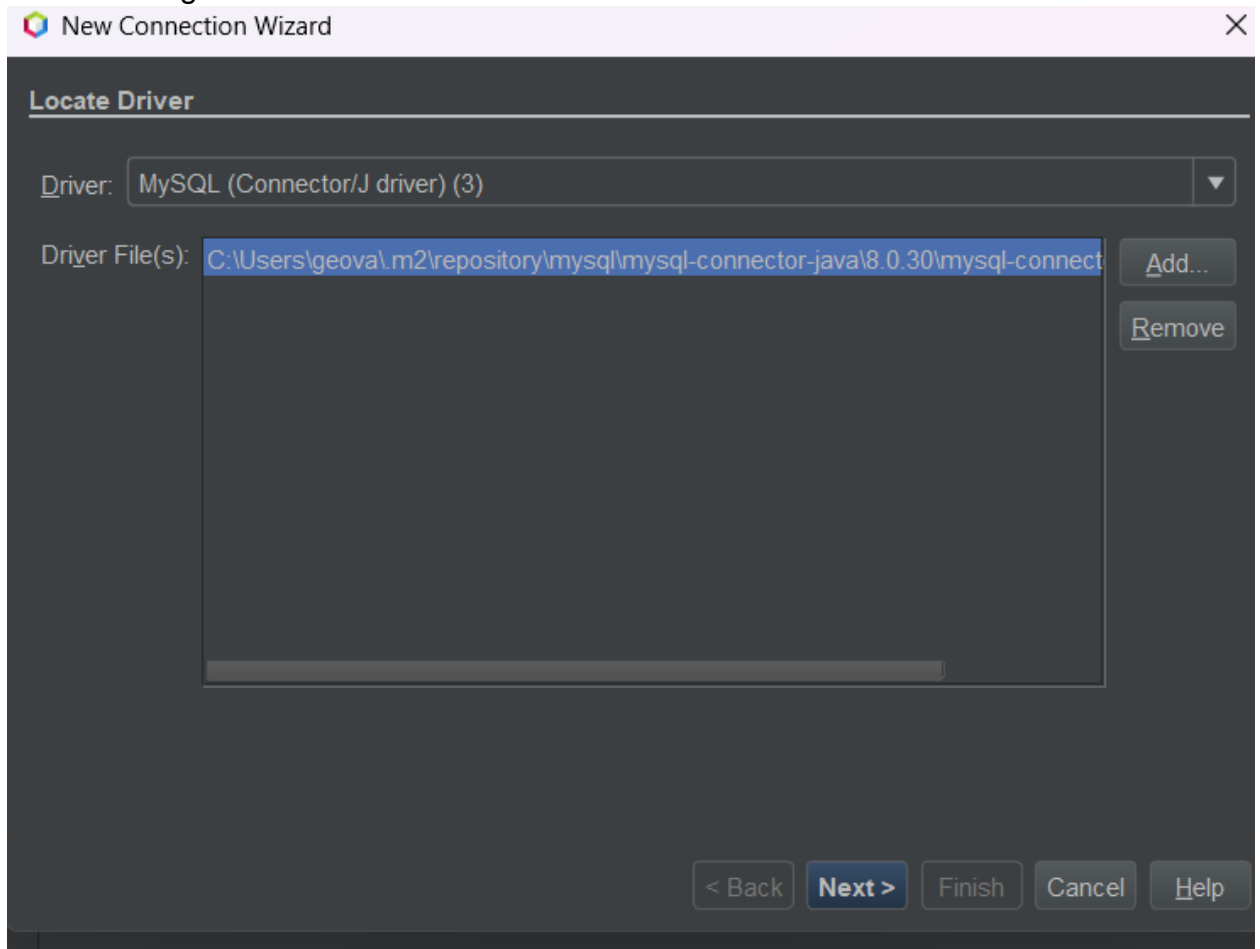
9. Ahora nos vamos a services – Databases – New Connection



10. Agregamos el driver que acabamos de descargar



11. Le damos siguiente



12. Llenamos datos

New Connection Wizard

**Customize Connection**

Driver Name: MySQL (Connector/J driver) on MySQL (Connector/J driver) (1) ▼

Host: localhost Port: 3306

Database: proyectoprograiii

User Name: Geovany

Password: .....  
☐ Remember password

Connection Properties Test Connection

JDBC URL: jdbc:mysql://localhost:3306/proyectoprograiii?serverTimezone=UTC

< Back Next > Finish Cancel Help

El serverTimezone nos sirve para que no hayan errores con los horarios.  
13. Avanzamos y le damos finish

New Connection Wizard

×

Choose name for connection

Override the default name for the connection. The name should be descriptive about the connection you are creating.

Input connection name:

jdbc:mysql://localhost:3306/proyectoPROGRAMAIII?serverTimezone=UTC [Geovany on Default schema]

< Back

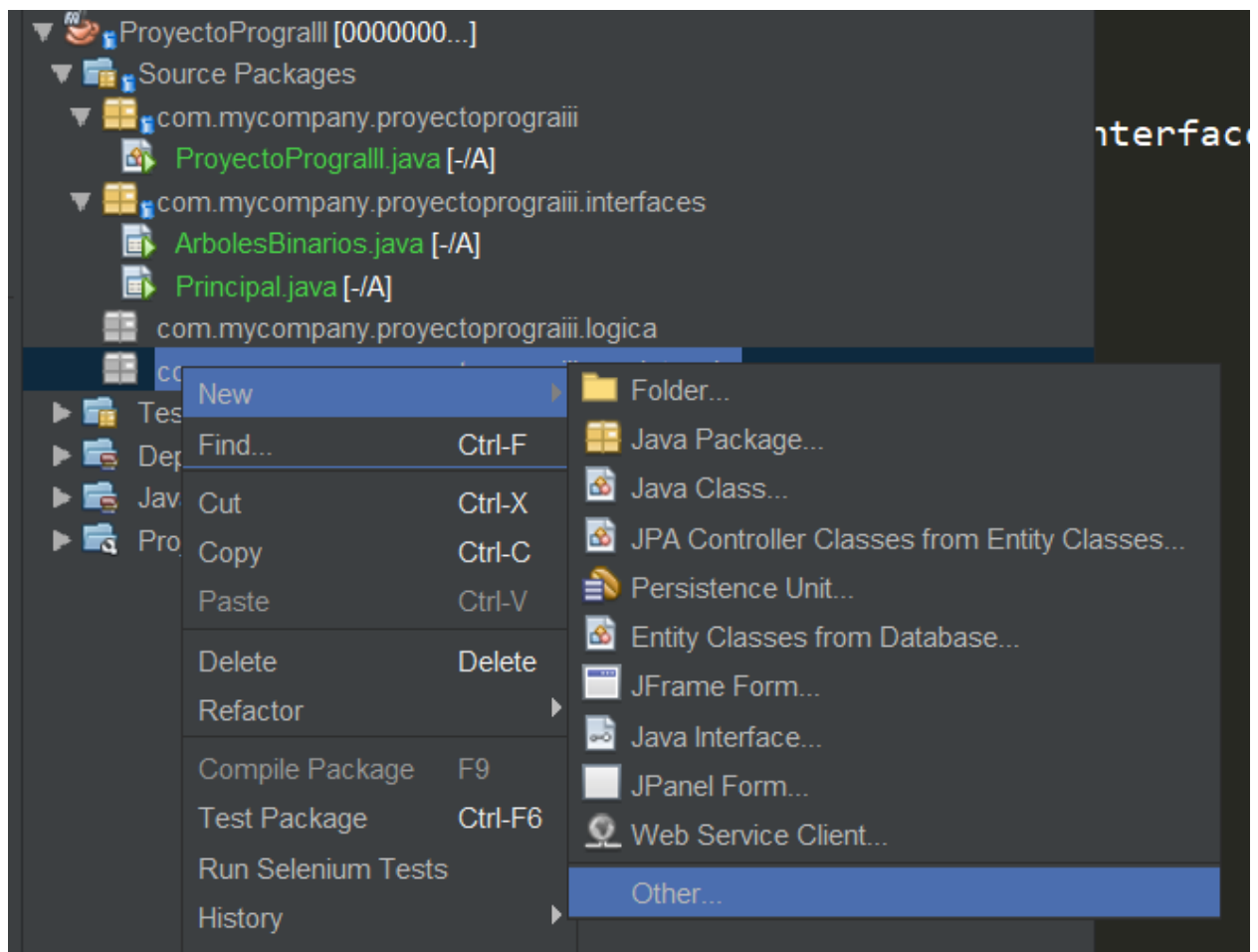
Next >

Finish

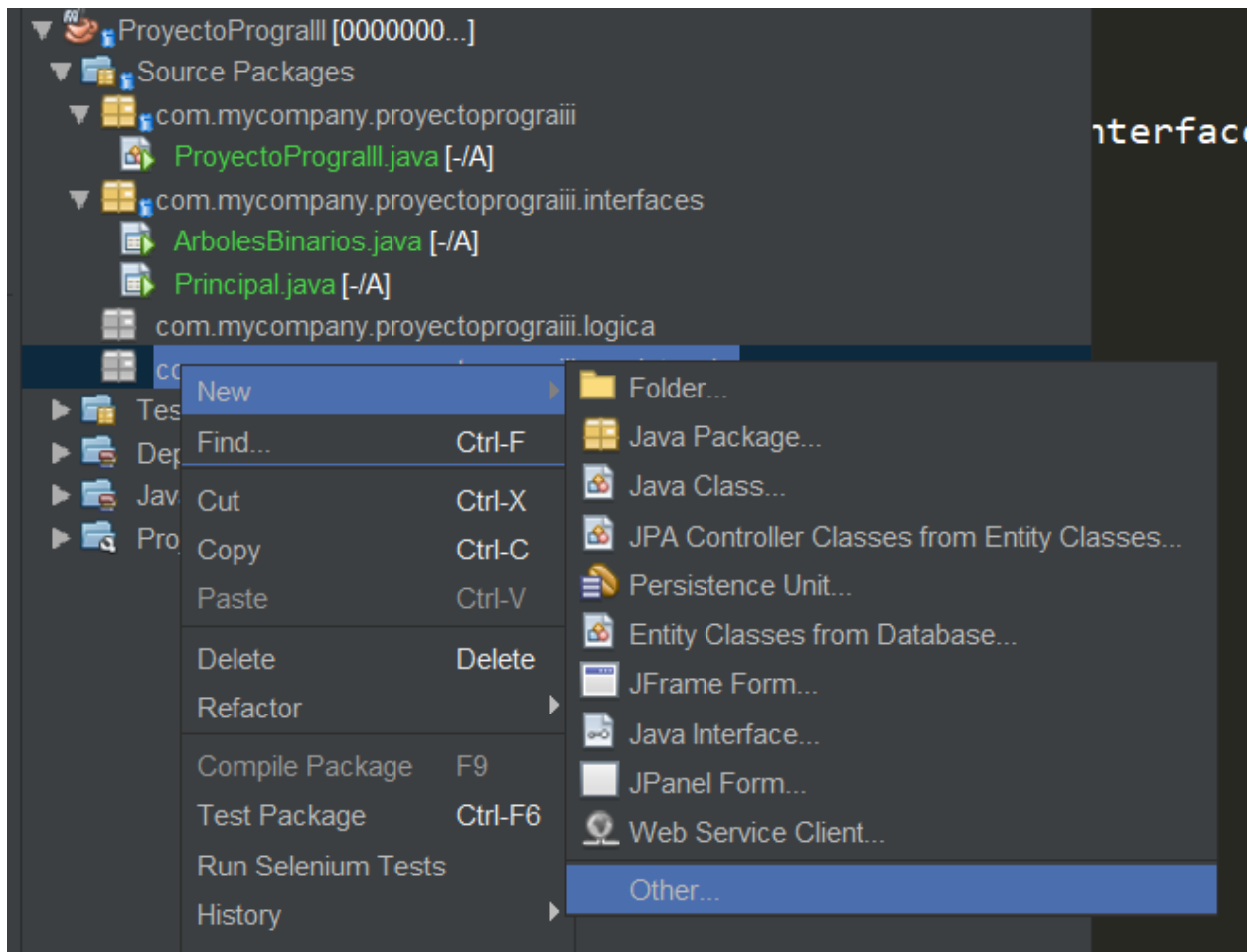
Cancel

Help

14. Ahora crearemos el archivo de persistence







15. Llenamos los datos

New Persistence Unit

**Steps**

1. Choose File Type
2. **Provider and Database**

**Provider and Database**

Persistence Unit Name:

Specify the persistence provider and database for entity classes.

Persistence Library:

Database Connection:

Table Generation Strategy: ☒ Create ☐ Drop and Create ☐ None

< Back Next > **Finish** Cancel Help

16. Se nos crea el persistence

Persistence Units

ProjectoprograPU

**General:**

Persistence Unit Name:

Persistence Library:

JDBC Connection:

☐ Use Java Transaction APIs

Table Generation Strategy: ☐ Create ☐ Drop and Create ☒ None

Validation Strategy: ☒ Auto ☐ Callback ☐ None

Shared Cache Mode: ☐ All ☐ None ☐ Enable Selective ☐ Disable Selective ☒ Unspecified

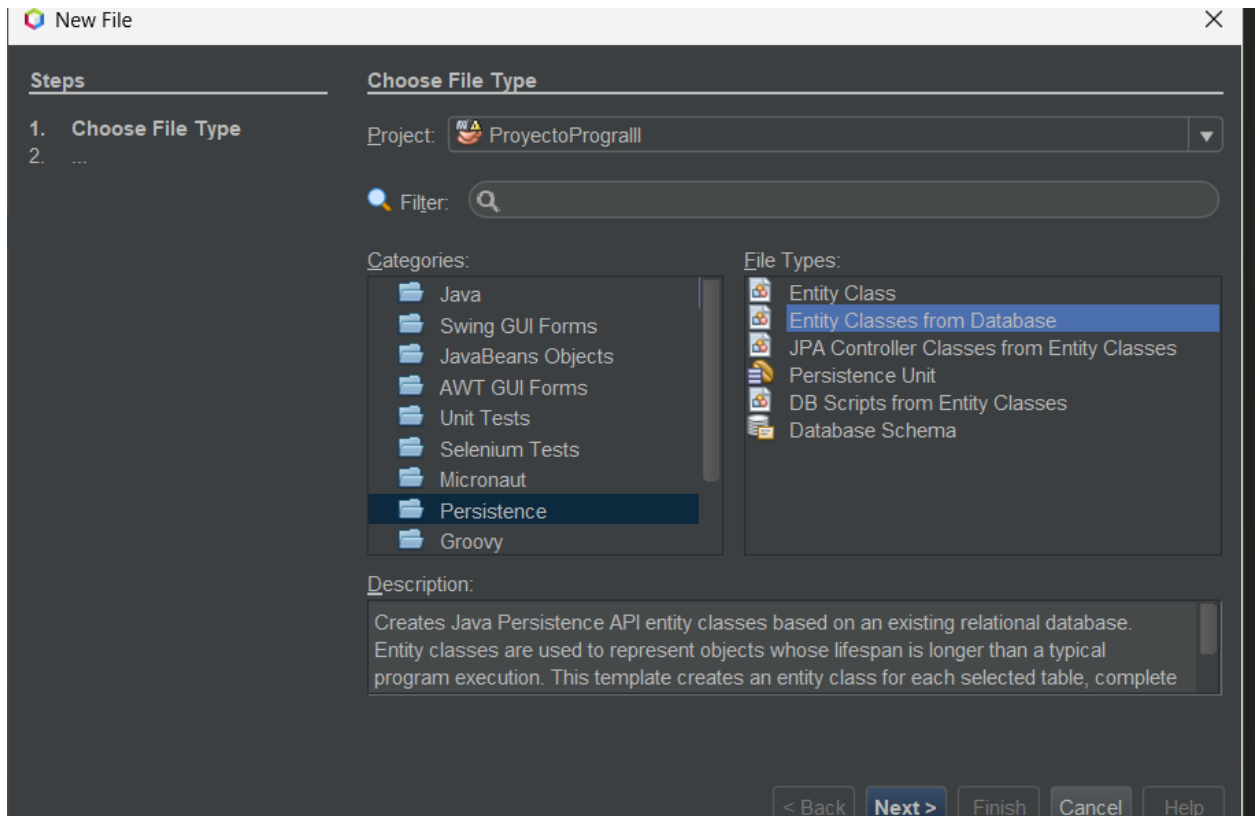
☐ Include All Entity Classes in "Projectoprograii" Module

Include Entity Classes

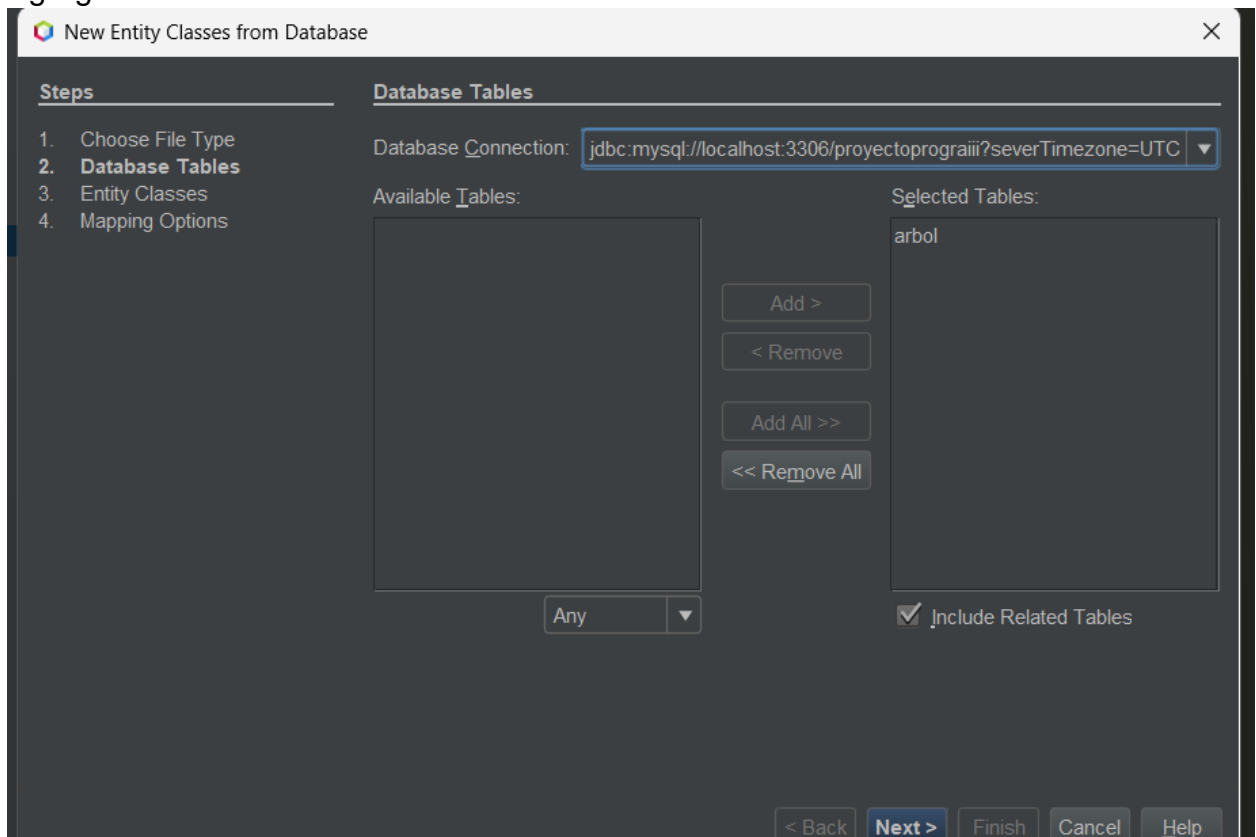
Add Class... Remove

**Properties:**

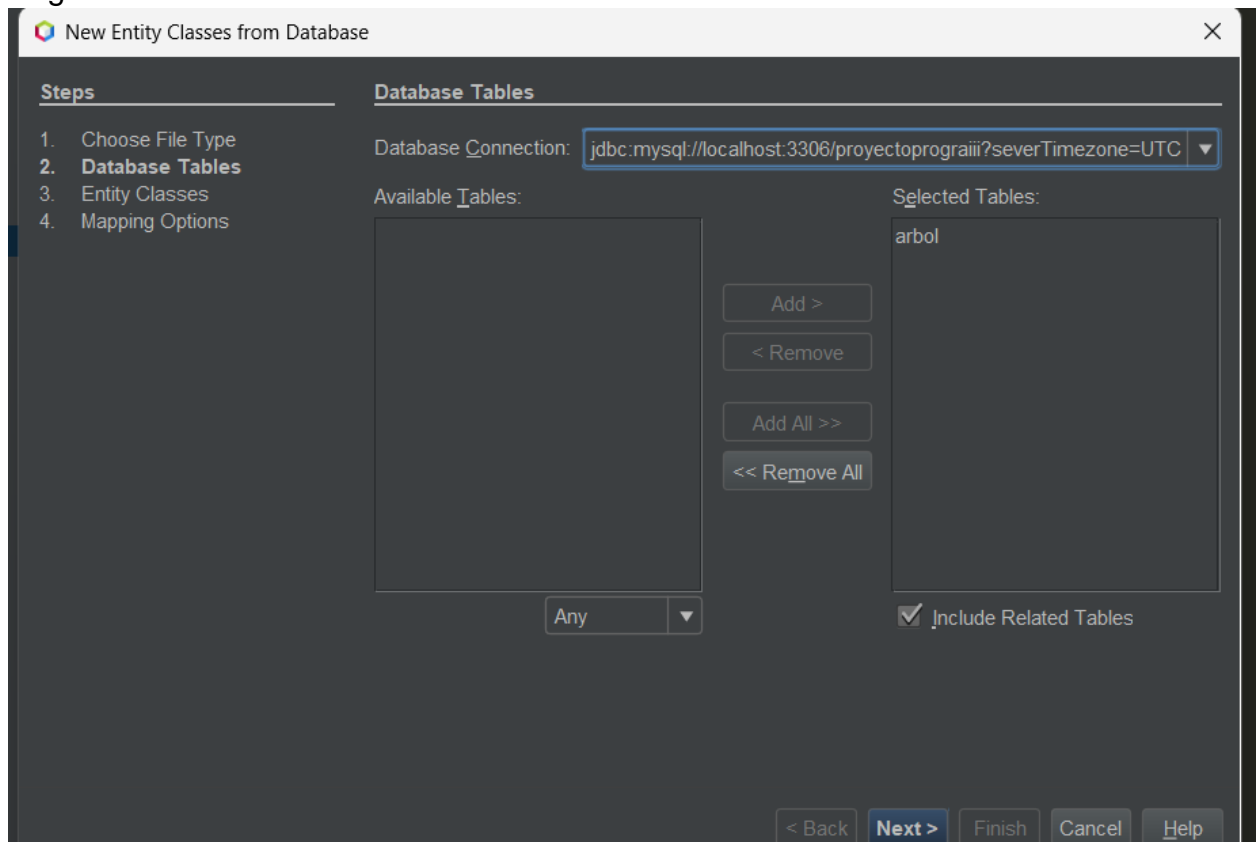
17. Crearemos la entity classes



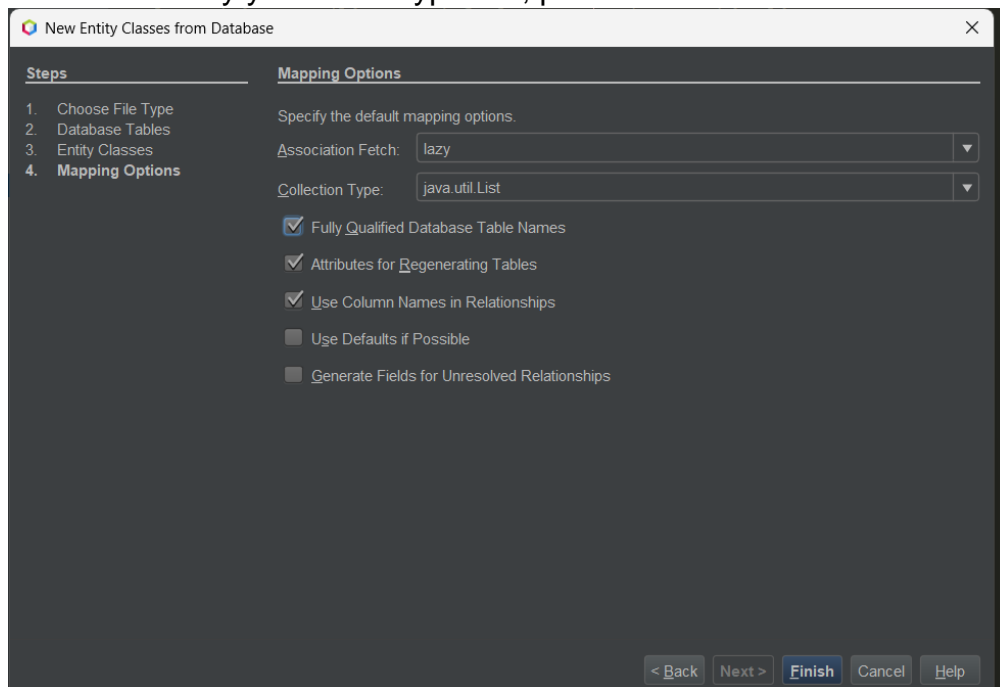
18. Agregamos la tabla.



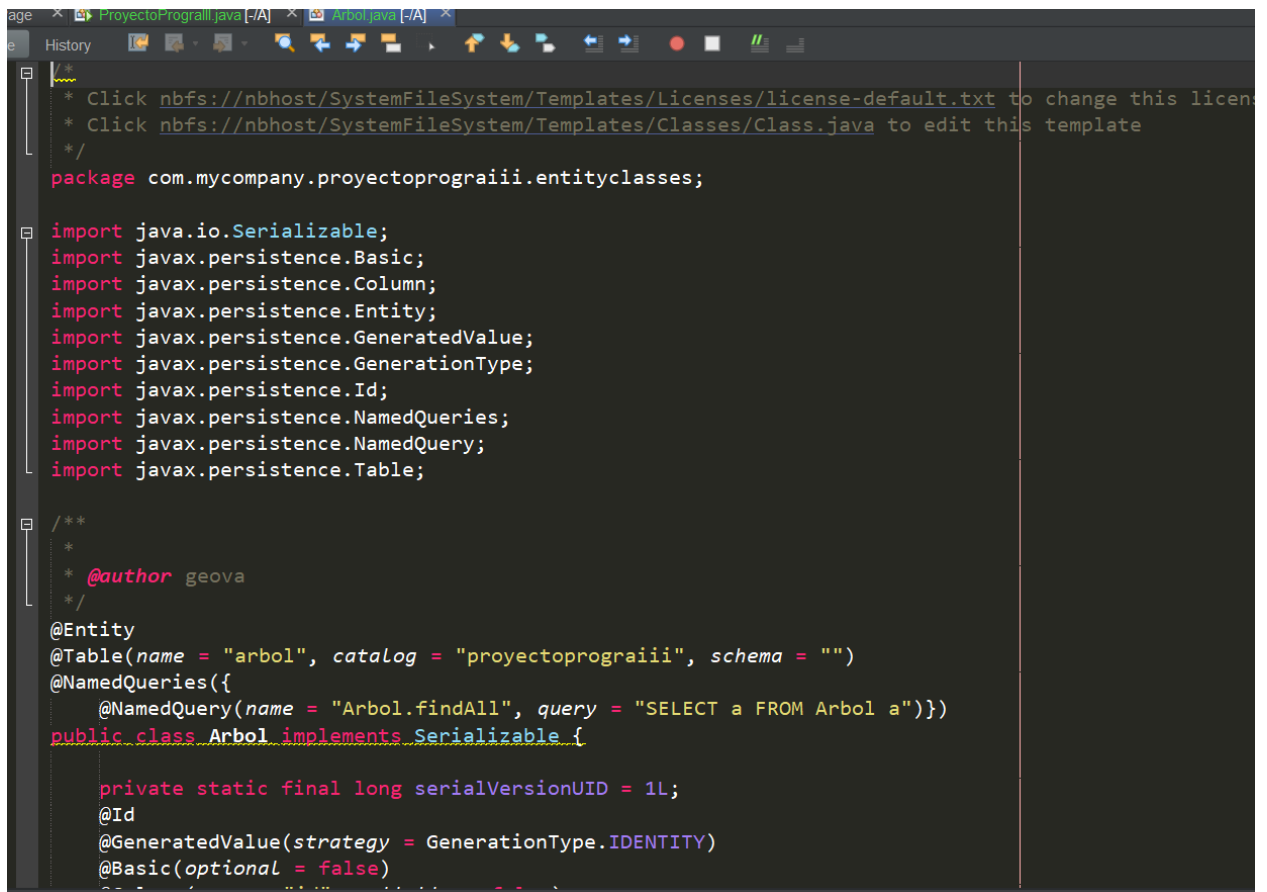
## 19. Seguimos avanzando



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



## 21. Ya la tenemos



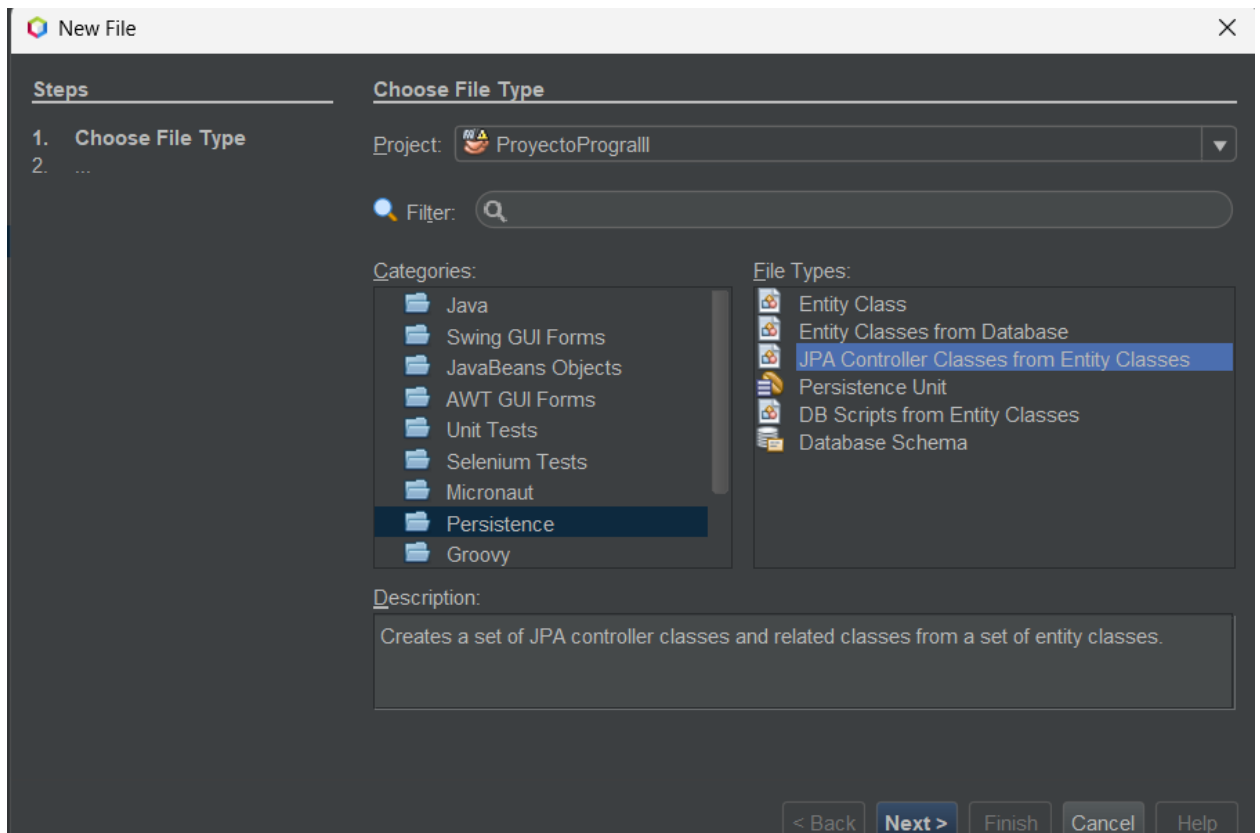
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.mycompany.proyectoprograiiii.entityclasses;

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;

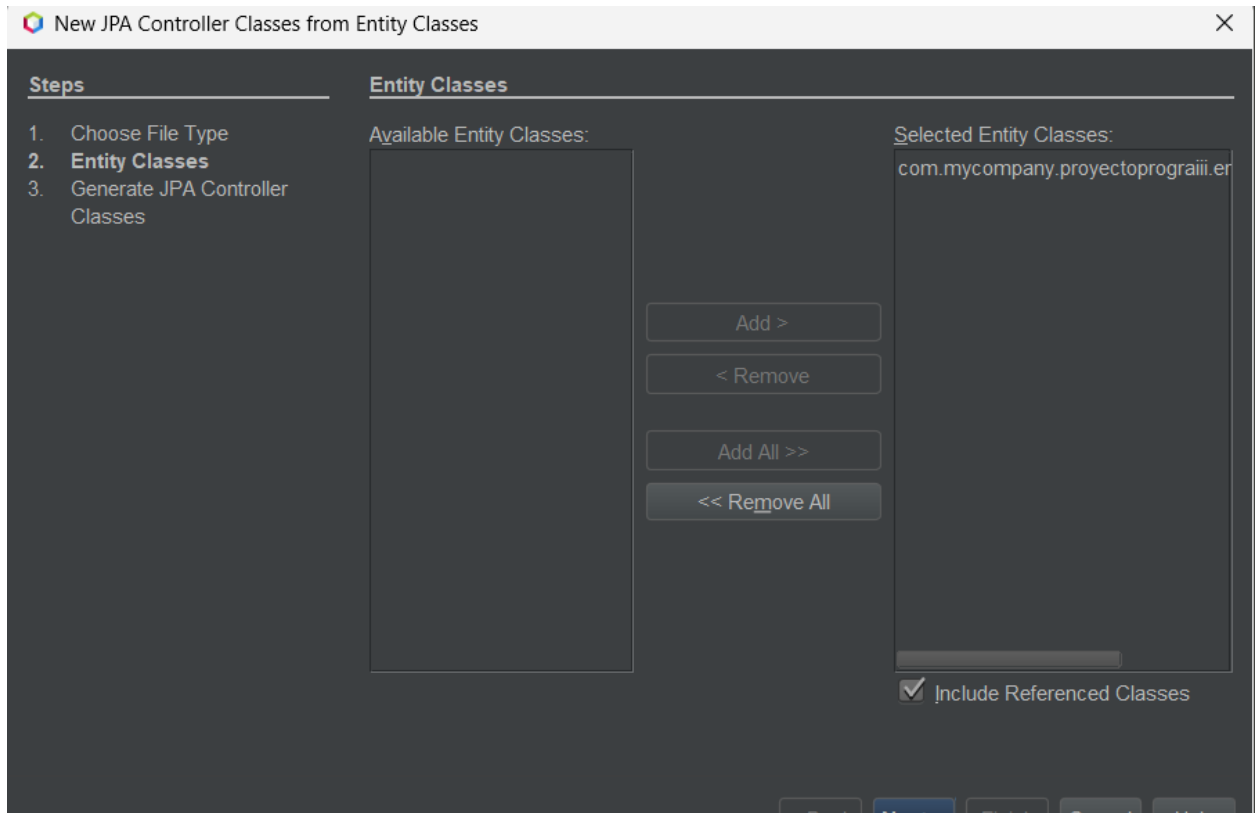
/**
 *
 * @author geova
 */
@Entity
@Table(name = "arbol", catalog = "proyectoprograiiii", schema = "")
@NamedQueries({
    @NamedQuery(name = "Arbol.findAll", query = "SELECT a FROM Arbol a")})
public class Arbol implements Serializable {

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

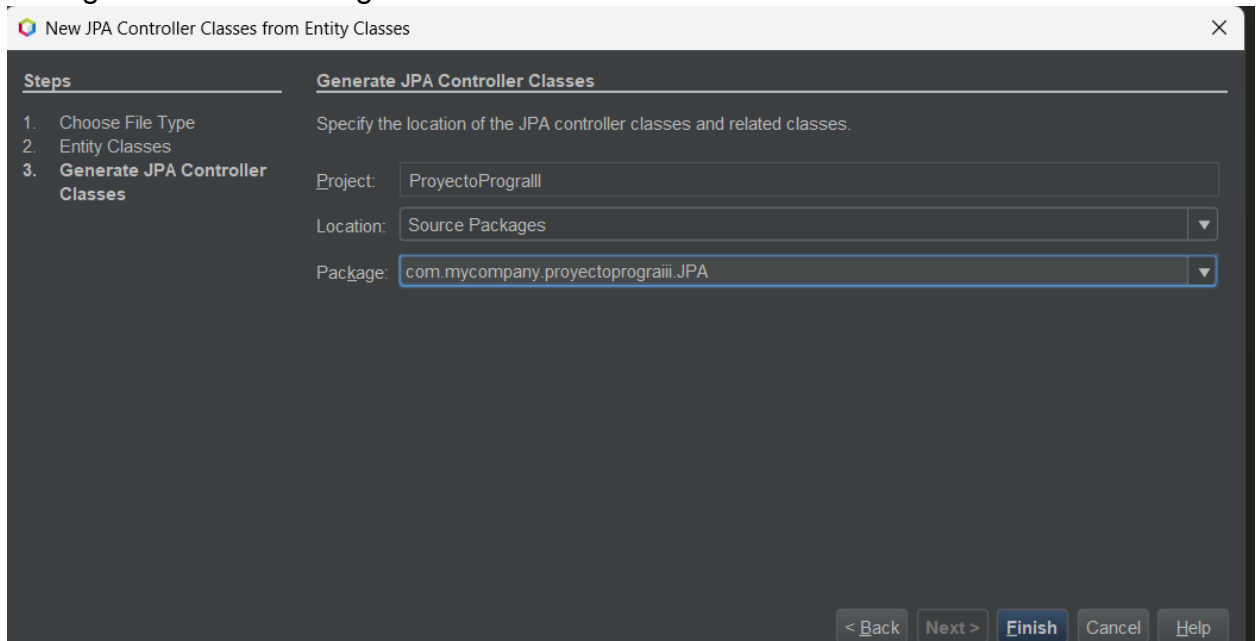
## 22. Ahora crearemos el jpa



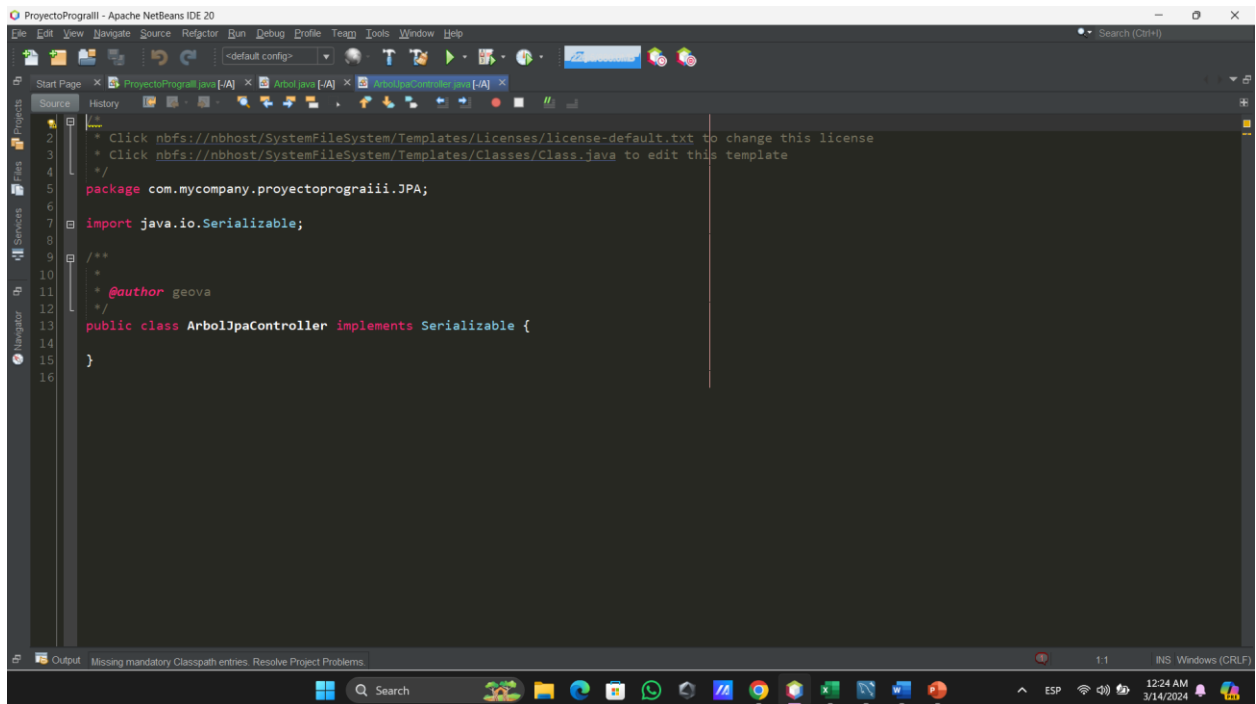
## 23. Pasamos las clases



## 24. Configuramos donde se guardará



## 25. Creado el JPA



## 26. Pegar este código

```
package com.mycompany.proyectoprograiii.JPAC;

import com.mycompany.proyectoprograiii.entityclasses.Arbol;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Query;
```

```

import java.util.List;
import javax.persistence.EntityManager;

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();
    }
}
```