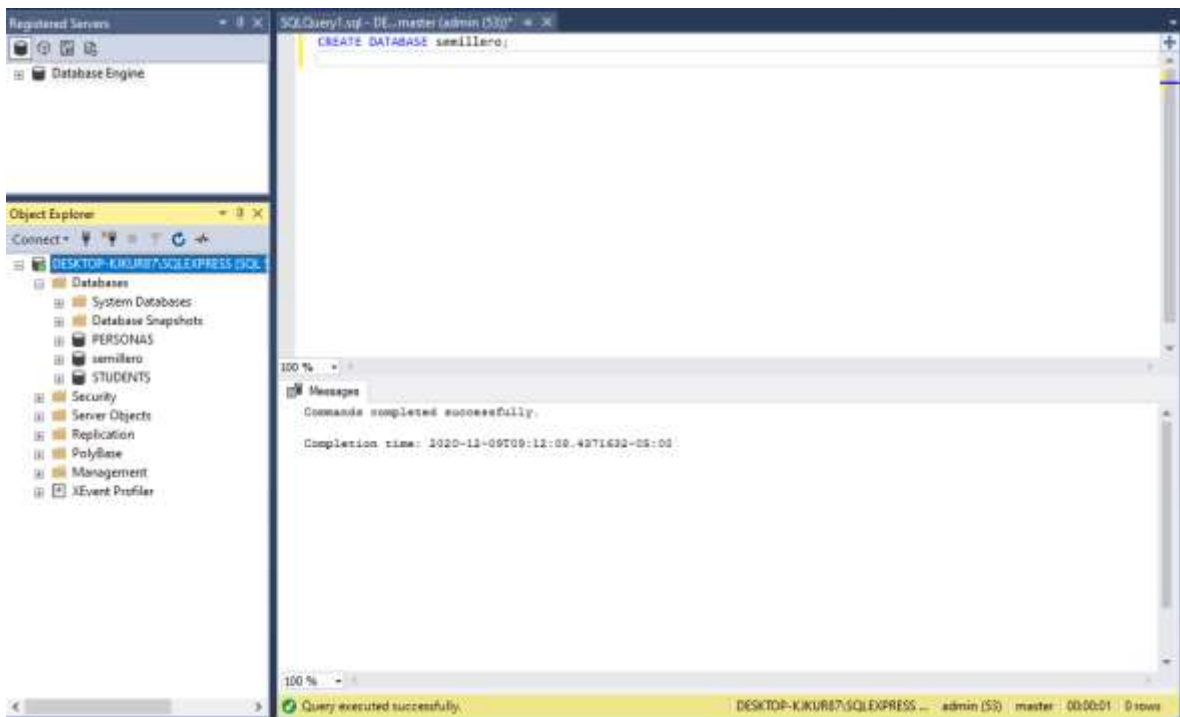
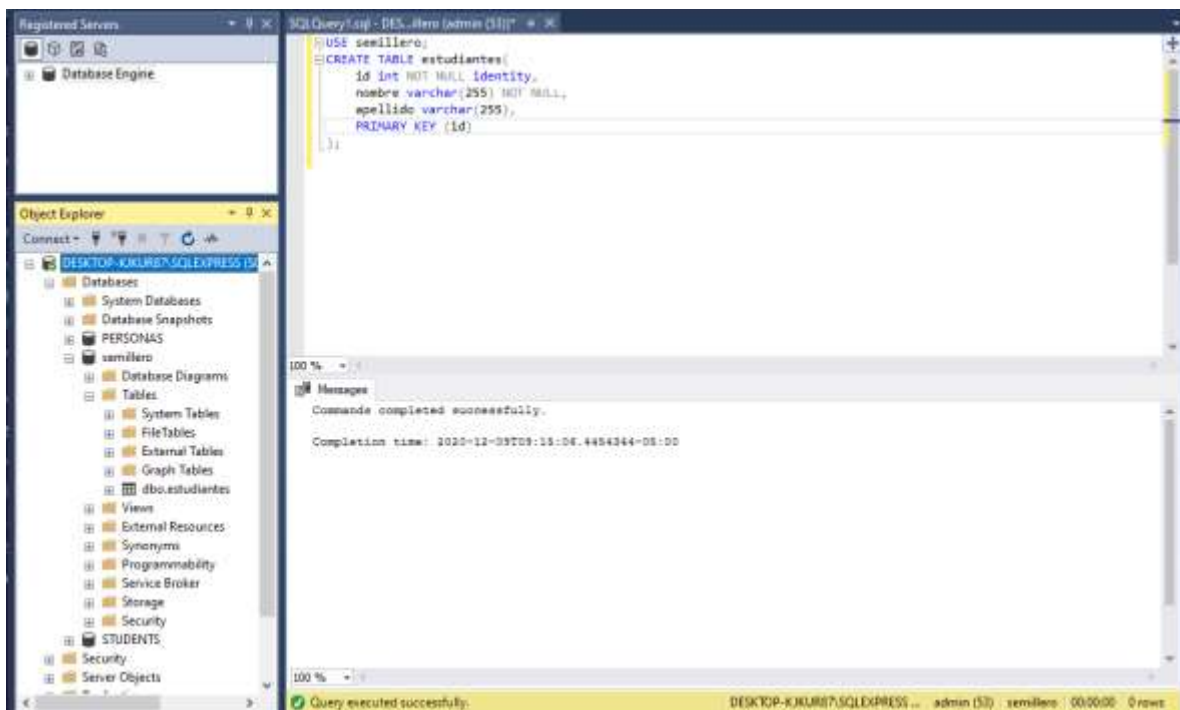


Crear una base de datos llamada semillero.



Crear una tabla llamada estudiantes, usando el siguiente script.



Insertar un registro en la base de datos.

```
use semillero;  
INSERT INTO estudiantes(nombre,apellido) values ('Santiago','Mosquera');
```

100 %

Messages

(1 row affected)

Completion time: 2020-12-09T09:23:17.0138719-05:00

Ejecute un select de la tabla estudiantes

```
SELECT * FROM estudiantes;
```

100 %

Results Messages

	id	nombre	apellido
1	1	Santiago	Mosquera

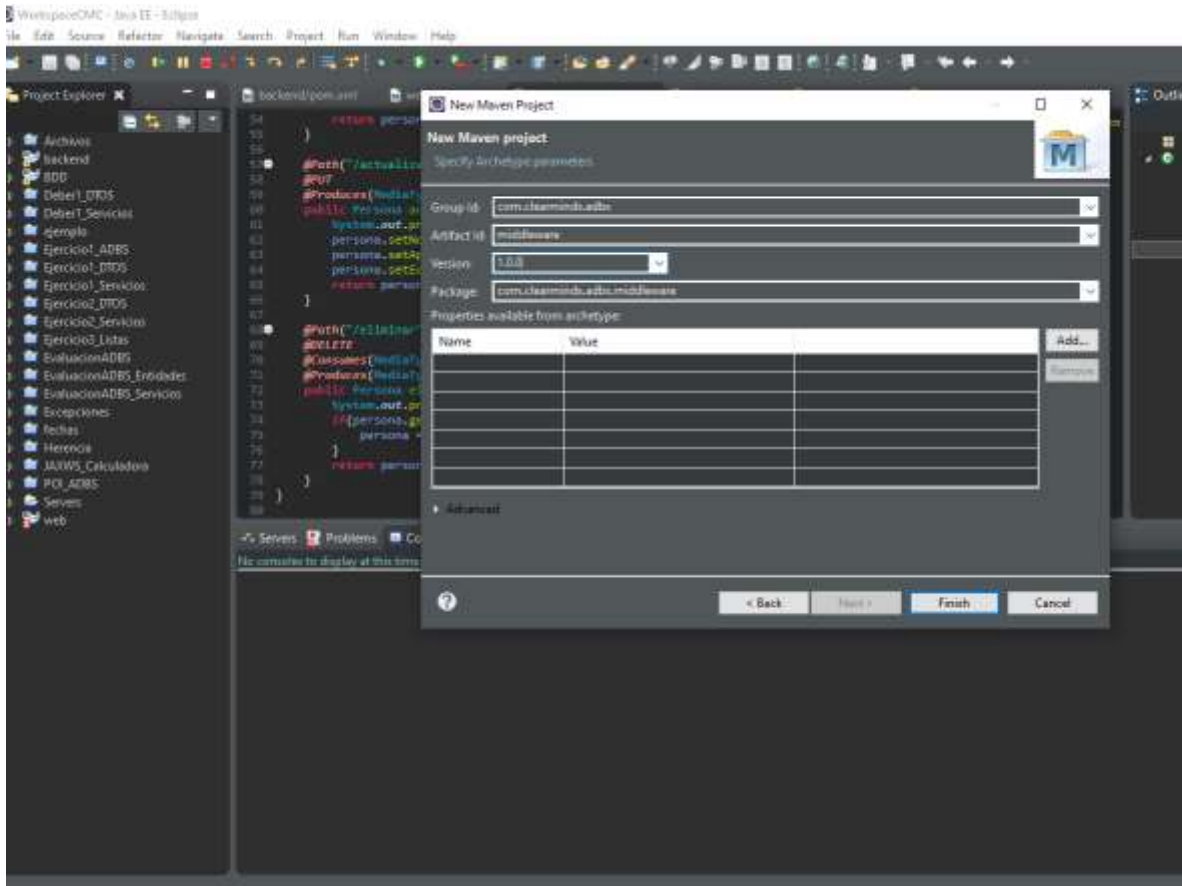
## Conexión BDD

Crear un proyecto maven, usando el archetype maven-archetype-quickstart con los datos:

groupId: com.clearminds.<SUS\_INICIALES>, por ejemplo: com.clearminds.smo

artifactId: middleware

versión:1.0.0




Crear un repositorio llamado tallerSemillero en su cuenta de github. El repositorio debe ser Public y agregar un .gitignore de Maven, para que ignore los archivos de maven que no deberíamos subir al repositorio.

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

Owner \*

 AndresBetancourt-Dev ▾

Repository name \*

/

tallerSemillero



Great repository names are short and memorable. Need inspiration? How about [shiny-octo-rotary-phone?](#)

Description (optional)



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

**Initialize this repository with:**

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)


☒ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: **Maven** ▾

☐ **Choose a license**


A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

**Create repository**

MODIFICAR el nombre de la rama principal a master, en lugar de main, presionando settings

New repositories you create will use 'master' as their default branch name.

**AndresBetancourt...**  
Personal settings

Profile

Account

Appearance New

Account security

Billing & plans

Security log

Security & analysis

Emails

Notifications

SSH and GPG keys

Repositories

Organizations

Saved replies

Applications


Developer settings


Moderation settings

## Repository default branch


Choose the default branch for your new personal repositories. You might want to change the default name to match your workflow, or because your integrations still require "master" as the default branch name. You can also change the default branch name on individual repositories. [Learn more about default branches.](#)


## Repositories


 Repositories


 Deleted repositories


**AndresBetancourt-Dev**


 AndresBetancourt-Dev/.NET-Blazor 409 KB 1 collaborator

 AndresBetancourt-Dev/.NET-First-Exam 19.5 MB 1 collaborator

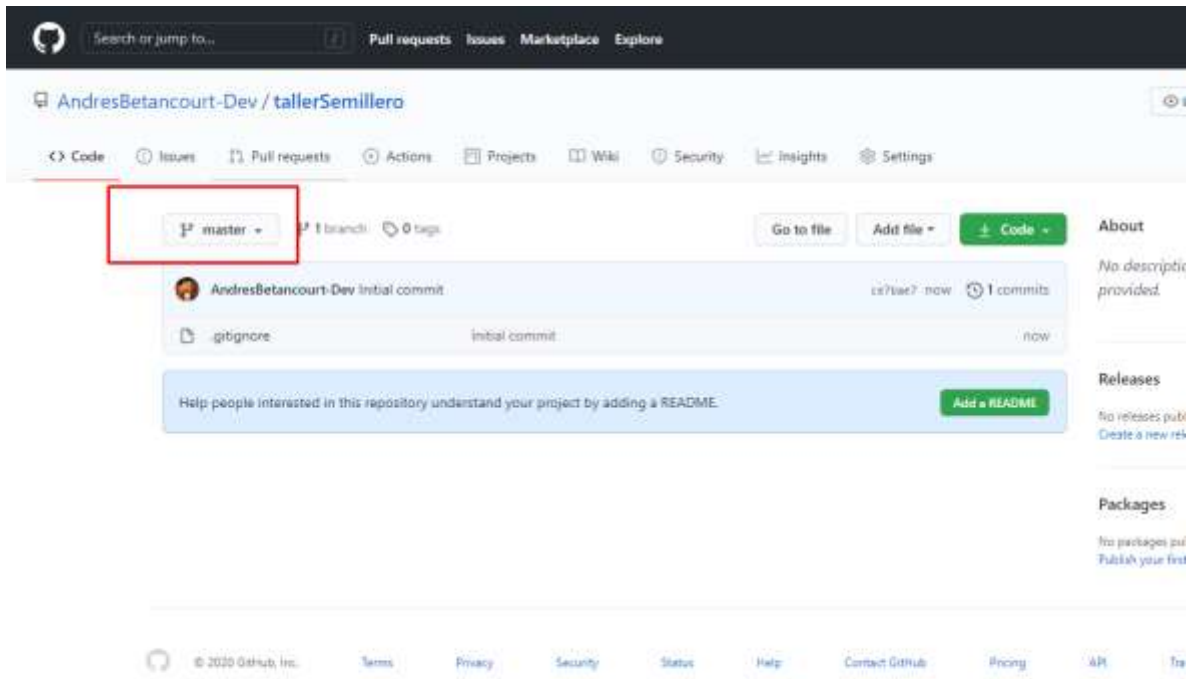
 AndresBetancourt-Dev/.NET-MONGO-REST 2.17 MB 0 collaborators

 AndresBetancourt-Dev/.NET-MVC 17.1 MB 1 collaborator

 AndresBetancourt-Dev/.NET-Razor-Pages 16.9 MB 0 collaborators

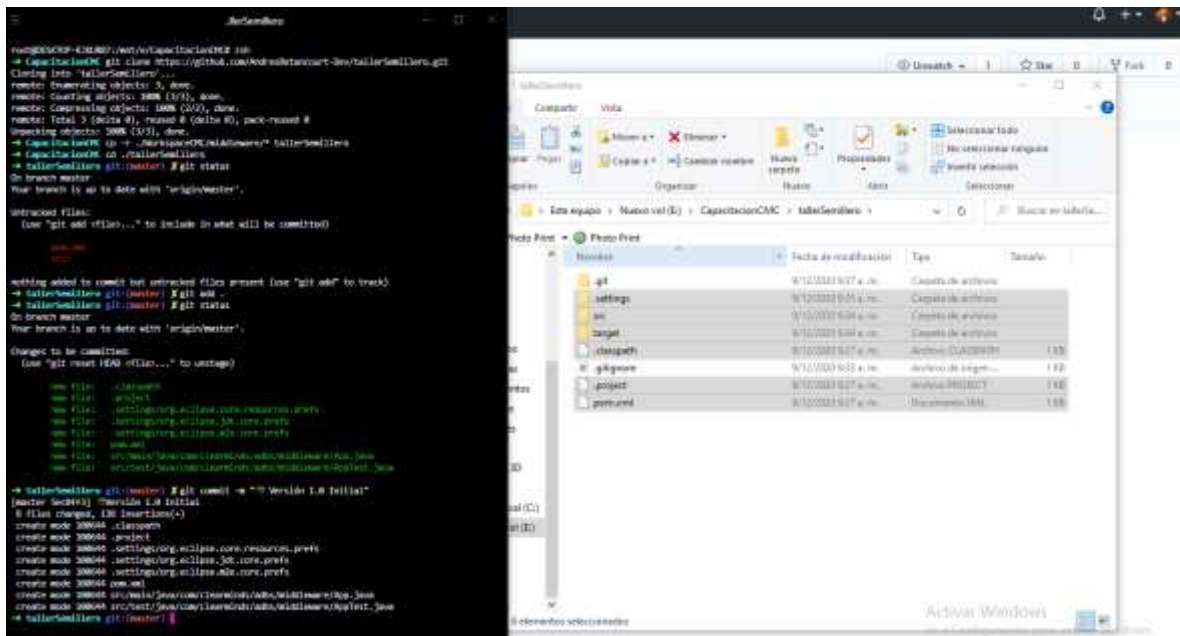
 AndresBetancourt-Dev/.NET-REST-API 10.1 MB 0 collaborators

Completar la creación del proyecto, y verificar que se creó con la rama master



Subir su proyecto recién creado al repositorio, para lo cual debe seguir los siguientes pasos:

1. En un directorio fuera del Workspace, clonar el repositorio usando el comando `git clone` y la URL del repositorio. Puede ubicar el URL del repositorio en CODE y copiarlo
2. Copiar los archivos del proyecto middleware en la carpeta clonada tallerSemillero
3. Dentro de tallerSemillero, en git Bash, ejecutar el comando `git status`
4. Ejecutar el comando `git add .`, para que todos los archivos untracked puedan ser manejados por maven
5. Nuevamente `git status`
6. Hacer un commit con el mensaje Versión inicial



1. Nuevamente git status
2. Subir los cambios al repositorio central usando git push

```

tallerSemillero
...llerSemillero

→ tallerSemillero git:(master) git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
→ tallerSemillero git:(master) git push
Username for 'https://github.com': AndresBetancourt-Dev
Password for 'https://AndresBetancourt-Dev@github.com':
Counting objects: 24, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (24/24), 2.73 KiB | 399.00 KiB/s, done.
Total 24 (delta 0), reused 0 (delta 0)
To https://github.com/AndresBetancourt-Dev/tallerSemillero.git
ce7bae7..5ec0ff3 master -> master
→ tallerSemillero git:(master) █

```

Git status

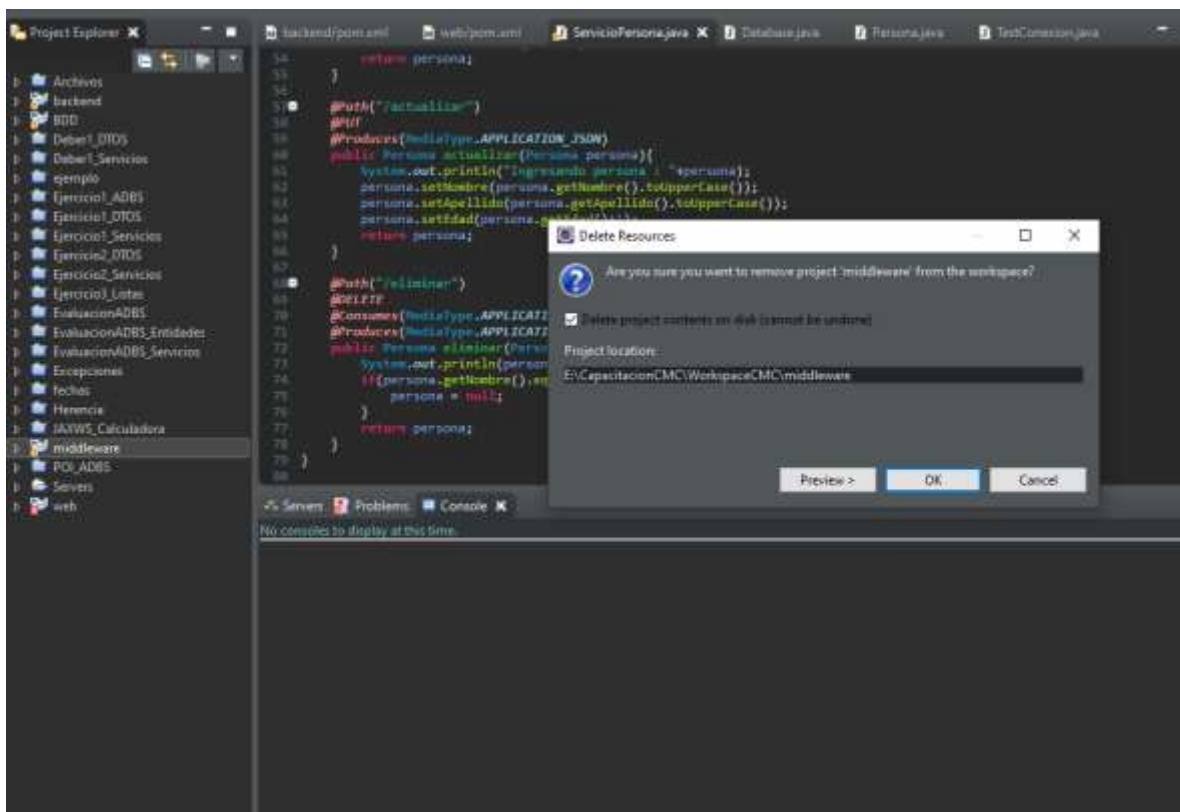
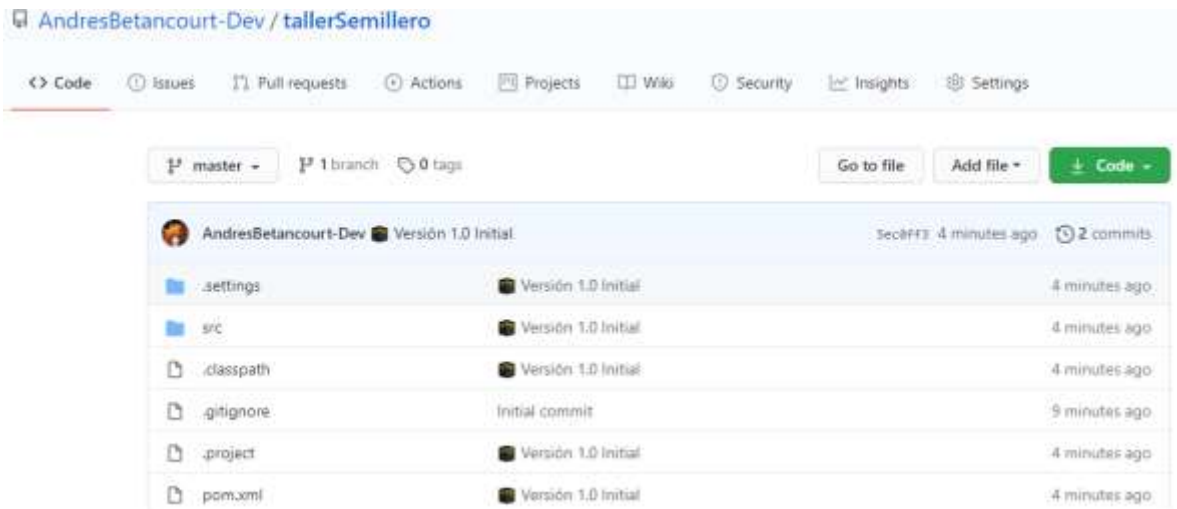
```

ce7bae7..5ec0ff3 master -> master
→ tallerSemillero git:(master) git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
→ tallerSemillero git:(master) █

```

1. Verificar en github, que se ha subido la versión inicial del proyecto.



1. Importar el proyecto Maven, desde la ubicación tallerSemillero



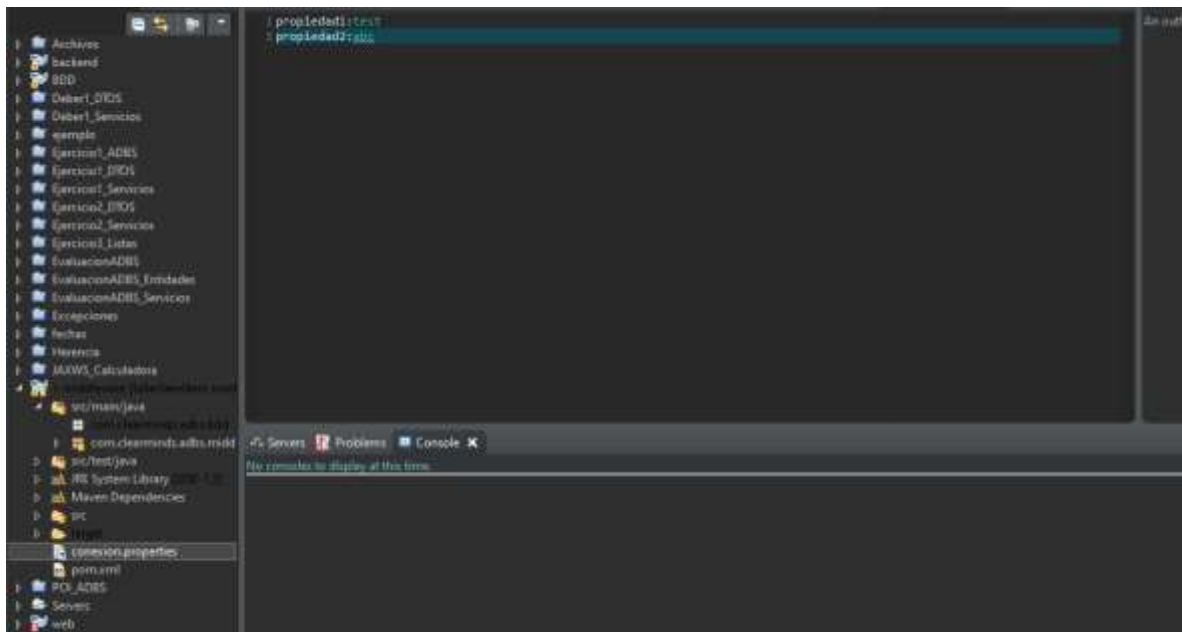
## Conexión BDD

En el proyecto middleware

Crear un paquete com.clearminds.<SUS\_INICIALES>.bdd

Dentro del paquete crear una clase ConexionBDD.

En la carpeta middleware, crear un archivo conexion.properties, con dos propiedades:



```

7 import java.util.Properties;
8
9 public class ConexionBDD {
10
11     public static void main(String[] args) {
12         LeerPropiedad("");
13     }
14
15     public static String leerPropiedad(String propiedad){
16         Properties properties = new Properties();
17         File file = new File("conexion.properties");
18
19         try {
20             properties.load(new FileReader(file));
21             System.out.println("uno="+properties.getProperty(propiedad));
22             return properties.getProperty(propiedad);
23
24         } catch (FileNotFoundException e) {
25             e.printStackTrace();
26         } catch (IOException e) {
27             e.printStackTrace();
28         }
29         return null;
30     }
31 }

```

```

32
33 public class ConexionBDD {
34
35     public static void main(String[] args) {
36         System.out.println(LeerPropiedad("propiedad1"));
37         System.out.println(LeerPropiedad("propiedad2"));
38         System.out.println(LeerPropiedad(""));
39     }
40
41     public static String leerPropiedad(String propiedad){
42         Properties properties = new Properties();
43         File file = new File("conexion.properties");
44         try {
45             properties.load(new FileReader(file));
46             return properties.getProperty(propiedad) != null ? properties.getProperty(propiedad) : null;
47
48         } catch (FileNotFoundException e) {
49             e.printStackTrace();
50         } catch (IOException e) {
51             e.printStackTrace();
52         }
53         return null;
54     }
55 }

```

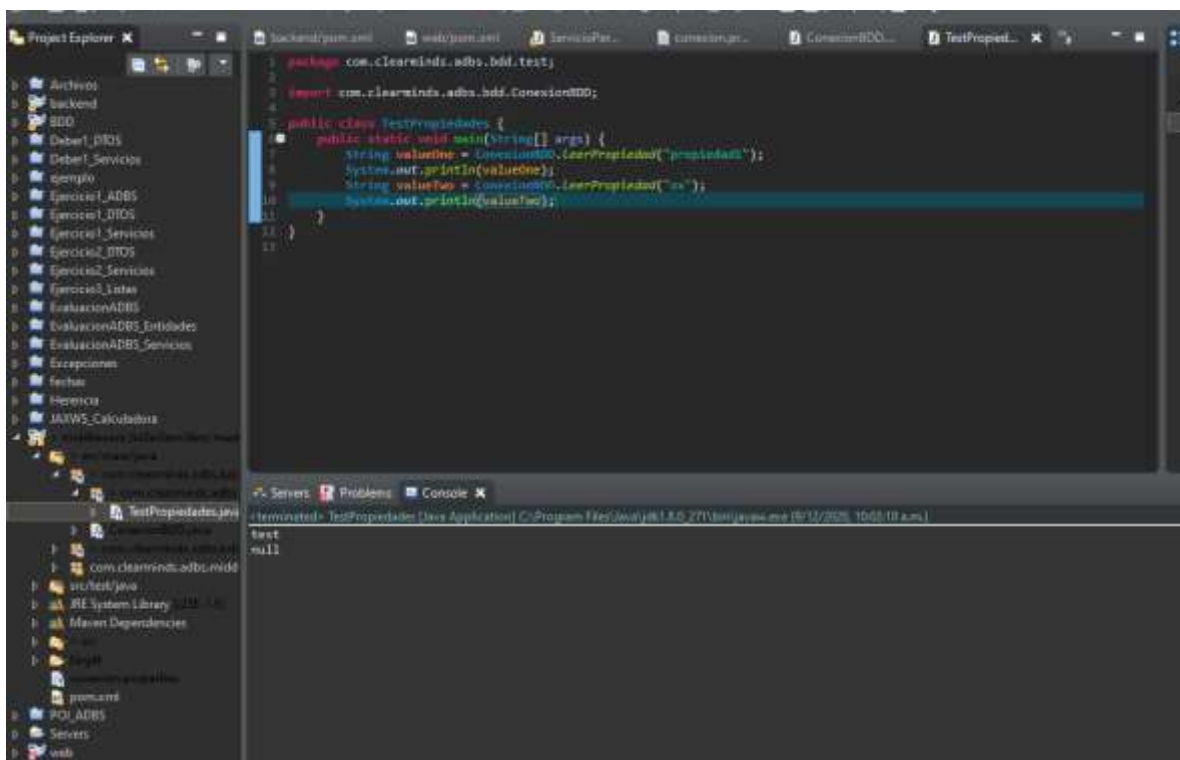
En caso de excepciones retornar null. Si la propiedad no existe, retornar null.

Crear un paquete com.clearminds.<SUS\_INICIALES>.bdd.test

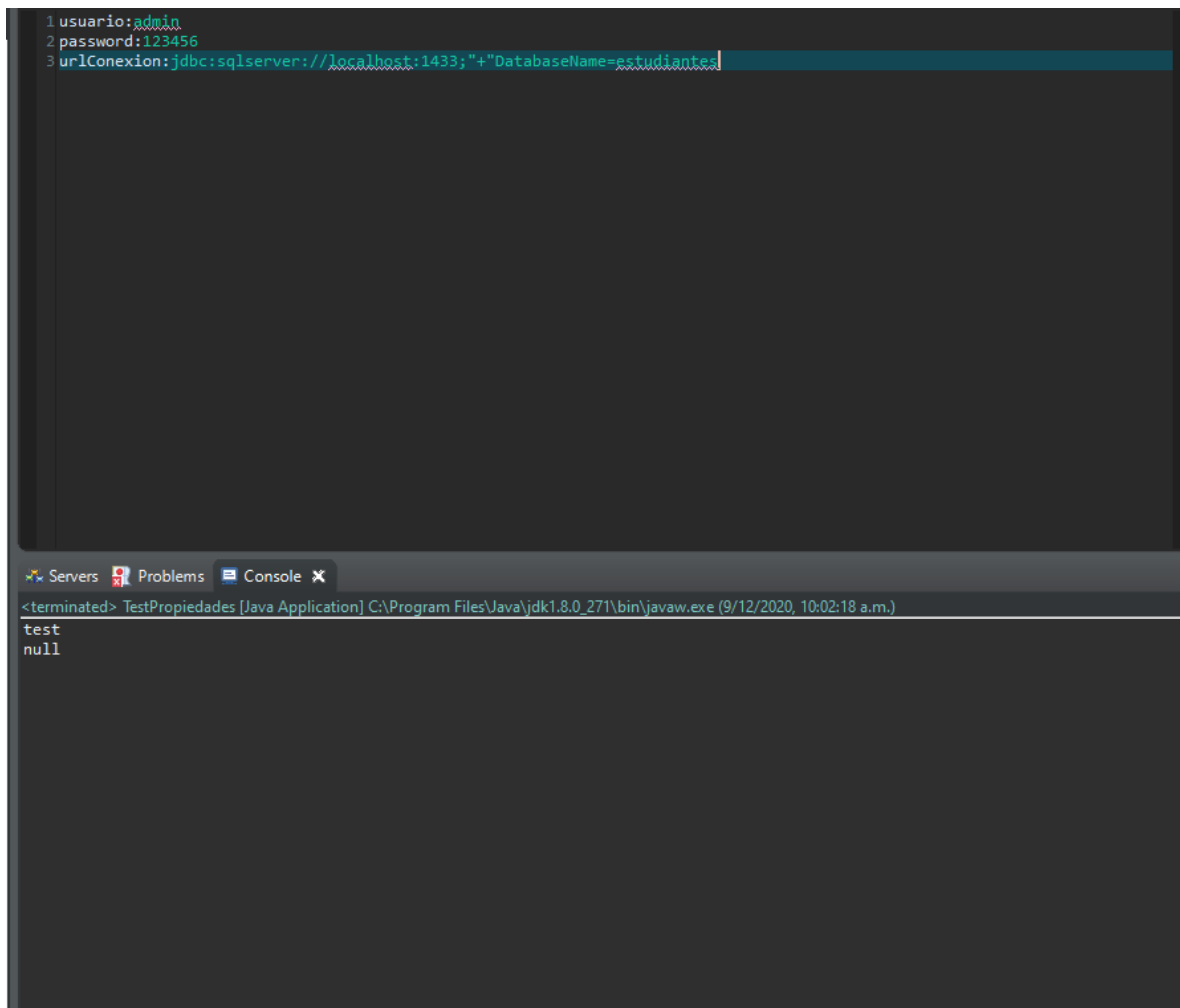
Crear una clase TestPropiedades, con un método main, probar el método leerPropiedad

**Crear una carpeta evidencias, subir al repositorio y en esta carpeta ir subiendo las capturas de pantallas de los Tests que se van pidiendo**

**SUBIR LOS FUENTES AL REPOSITORIO**



Modificar el archivo de propiedades, borrar las actuales y agregar las propiedades con los datos correspondientes a su base de datos:



The screenshot shows an IDE with two panels. The top panel displays a properties file with the following content:

```
1 usuario:admin
2 password:123456
3 urlConexion:jdbc:sqlserver://localhost:1433;"+"DatabaseName=estudiantes
```

The bottom panel shows the console output for a Java application named 'TestPropiedades'. The output is:

```
<terminated> TestPropiedades [Java Application] C:\Program Files\Java\jdk1.8.0_271\bin\javaw.exe (9/12/2020, 10:02:18 a.m.)
test
null
```

Agregar un método estático llamado `obtenerConexion`, que no recibe parámetros, retorna `Connection` y ejecuta la siguiente lógica:

- 1) Lee cada una de las propiedades desde el archivo de propiedades y las guarda en variables, por ejemplo  
`String url=leerPropiedad("urlConexion");`

```

public static Connection obtenerConexion() throws BDDEException{
    Connection connection = null;
    String url = LeerPropiedad("urlConexion");
    String usuario = LeerPropiedad("usuario");
    String password = LeerPropiedad("password");
    try{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        connection = DriverManager.getConnection(url, usuario, password);
    }catch(Exception e){
        e.printStackTrace();
        throw new BDDEException("No se pudo Conectar a la base de datos.");
    }
    return connection;
}

```

- 2) Usando las propiedades recuperadas, invoca al método estático getConnection de DriverManager y retorna el Connection obtenido.

En caso de excepciones, manejarlo de la siguiente forma:

```

public static Connection obtenerConexion() throws BDDEException{
    Connection connection = null;
    String url = LeerPropiedad("urlConexion");
    String usuario = LeerPropiedad("usuario");
    String password = LeerPropiedad("password");
    try{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        connection = DriverManager.getConnection(url, usuario, password);
    }catch(Exception e){
        e.printStackTrace();
        throw new BDDEException("No se pudo Conectar a la base de datos.");
    }
    return connection;
}

```

Crear un nuevo paquete: com.clearminds.<SUS\_INICIALES>.excepciones, dentro de este paquete, crear una clase BDDEException de tipo Checked, que tenga un constructor que recibe el mensaje de la Excepción y con este valor invoca al constructor del padre Si ocurre una excepción, se lanza BDDEException con el mensaje: No se pudo conectar a la base de datos.

```

1 package com.clearminds.adbs.excepciones;
2
3 public class BDDEException extends Exception {
4
5     public BDDEException(String message) {
6         super(message);
7     }
8
9
10 }
11

```

- 1) Crear una clase TestConexion y probar el método obtenerConexion

```
1 package com.clearminds.adbs.bdd.test;
2
3 import java.sql.Connection;
4
5 import com.clearminds.adbs.bdd.ConexionBDD;
6 import com.clearminds.adbs.excepciones.BDDException;
7
8 public class TestConexion {
9     public static void main(String[] args) {
10         try{
11             Connection connection = ConexionBDD.obtenerConexion();
12             if(connection != null){
13                 System.out.println("Conexión Exitosa");
14             }
15         }catch(BDDException e){
16             e.printStackTrace();
17             System.out.println(e.getMessage());
18         }
19     }
20 }
21
```

Servers Problems Console

<terminated> TestConexion (1) [Java Application] C:\Program Files\Java\jdk1.8.0\_271\bin\javaw.exe (9/12/2020, 10:24:21 a.m.)

java.lang.ClassNotFoundException: com.microsoft.sqlserver.jdbc.SQLServerDriver  
at java.net.URLClassLoader.findClass(URLClassLoader.java:382)  
at java.lang.ClassLoader.loadClass(ClassLoader.java:418)  
at sun.misc.Launcher\$AppClassLoader.loadClass(Launcher.java:355)  
at java.lang.ClassLoader.loadClass(ClassLoader.java:351)  
at java.lang.Class.forName0(Native Method)  
at java.lang.Class.forName(Class.java:264)  
at com.clearminds.adbs.bdd.ConexionBDD.obtenerConexion(ConexionBDD.java:23)  
at com.clearminds.adbs.bdd.test.TestConexion.main(TestConexion.java:11)  
com.clearminds.adbs.excepciones.BDDException: No se pudo Conectar a la base de datos.  
at com.clearminds.adbs.bdd.ConexionBDD.obtenerConexion(ConexionBDD.java:27)  
at com.clearminds.adbs.bdd.test.TestConexion.main(TestConexion.java:11)  
No se pudo Conectar a la base de datos.

```
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>mssql-jdbc</artifactId>
<version>7.2.2.jre8</version>
</dependency>
```

```
1 package com.clearminds.adbs.bdd.test;
2
3 import java.sql.Connection;
4
5 import com.clearminds.adbs.bdd.ConexionBDD;
6 import com.clearminds.adbs.excepciones.BDDException;
7
8 public class TestConexion {
9     public static void main(String[] args) {
10         try{
11             Connection connection = ConexionBDD.obtenerConexion();
12             if(connection != null){
13                 System.out.println("Conexión Exitosa");
14             }
15         }catch(BDDException e){
16             e.printStackTrace();
17             System.out.println(e.getMessage());
18         }
19     }
20 }
21
```

Problems Javadoc Declaration Console Servers

<terminated> TestConexion (1) [Java Application] C:\Program Files\Java\jdk1.8.0\_271\bin\javaw.exe (9/12/2020, 10:26:21 a.m.)

Conexión Exitosa

- 1) Crear un nuevo paquete llamado com.clearminds.<SUS\_INICIALES>.servicios
- 2) Crear una clase llamada ServicioBase  
En ServicioBase, crear un atributo de tipo Connection llamado conexión  
Crear un método abrirConexion, que no recibe parámetros y no retorna nada. Este método se encarga de invocar al método obtenerConexion y asigna la conexión obtenida al atributo conexión.  
No hace try/catch de BDDException, sino que la propaga
- 3) Crear un método cerrarConexion, no recibe y no retorna, se encarga de cerrar la conexión usando el método close de Connection. Luego de cerrar la conexión imprime en consola "Conexión cerrada", antes de invocar a close valida si el atributo conexión es diferente de null. En caso de excepción solamente imprime el stackTrace y además imprime un mensaje en consola "Error al cerrar la conexion"
- 4) Crear la clase ServicioEstudiante, que hereda de ServicioBase, por concepto de herencia hereda los métodos abrirConexion y cerrarConexion

```

6  import com.clearminds.adbs.bdd.ConexionBDD;
7  import com.clearminds.adbs.excepciones.BDDException;
8
9  public class ServicioBase {
10     Connection conexion;
11
12     public void abrirConexion() throws BDDException{
13         conexion = ConexionBDD.obtenerConexion();
14     }
15
16     public void cerrarConexion(){
17         try {
18             if(conexion!=null){
19                 conexion.close();
20                 System.out.println("Conexión cerrada.");
21             }
22         } catch (Exception e) {
23             e.printStackTrace();
24             System.out.println("Error al cerrar la conexión.");
25         }
26     }
27 }
28

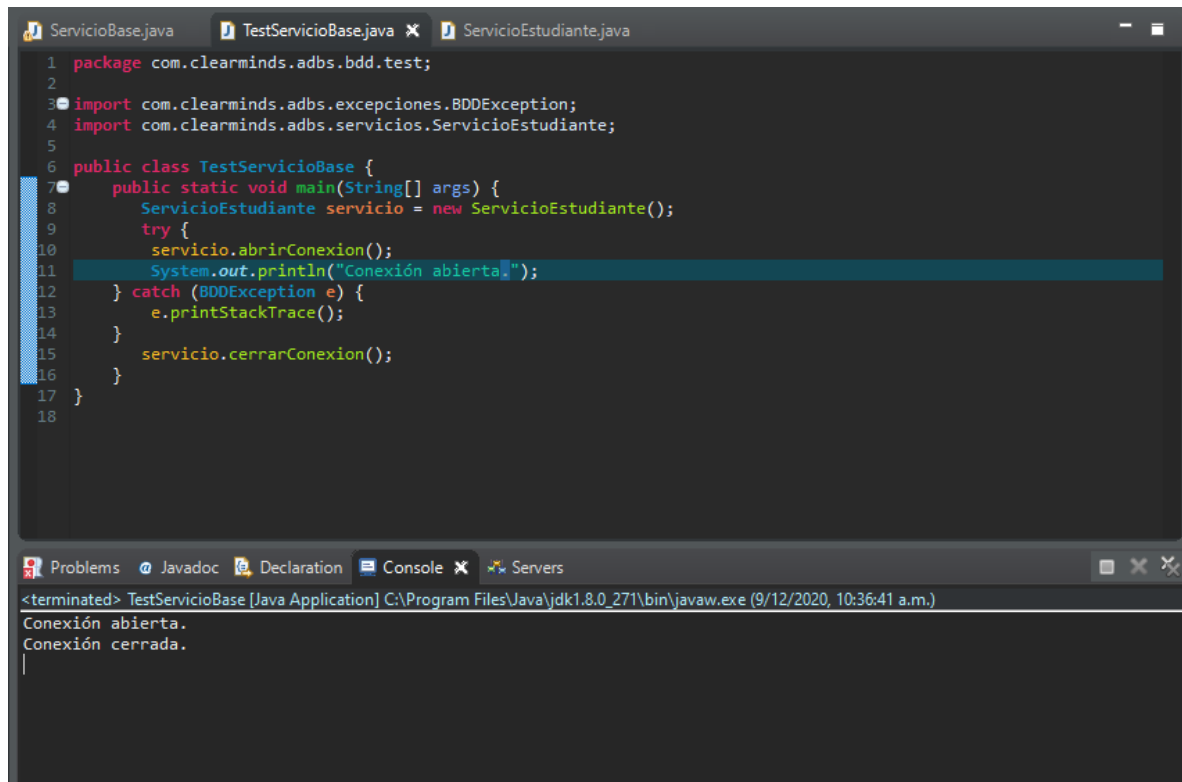
```

```

1  package com.clearminds.adbs.servicios;
2
3  public class ServicioEstudiante extends ServicioBase {
4
5  }
6

```





The image shows a screenshot of an IDE with three tabs: `ServicioBase.java`, `TestServicioBase.java`, and `ServicioEstudiante.java`. The `TestServicioBase.java` tab is active, displaying the following Java code:

```
1 package com.clearminds.adbs.bdd.test;
2
3 import com.clearminds.adbs.excepciones.BDDException;
4 import com.clearminds.adbs.servicios.ServicioEstudiante;
5
6 public class TestServicioBase {
7     public static void main(String[] args) {
8         ServicioEstudiante servicio = new ServicioEstudiante();
9         try {
10             servicio.abrirConexion();
11             System.out.println("Conexión abierta.");
12         } catch (BDDException e) {
13             e.printStackTrace();
14         }
15         servicio.cerrarConexion();
16     }
17 }
18
```

Below the code editor, the `Console` tab is active, showing the output of the program:

```
<terminated> TestServicioBase [Java Application] C:\Program Files\Java\jdk1.8.0_271\bin\javaw.exe (9/12/2020, 10:36:41 a.m.)
Conexión abierta.
Conexión cerrada.
|
```