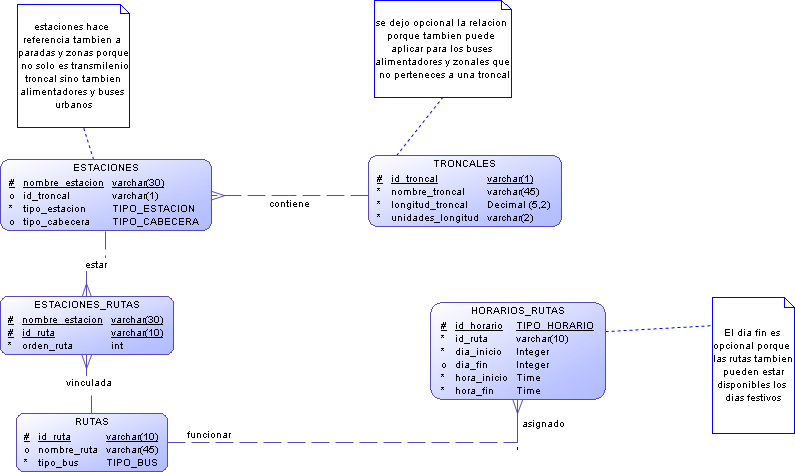
**PROYECTO TRANSMILENIO PARA EL MODULO DE FRAMEWORK DE PERSISTENCIA**

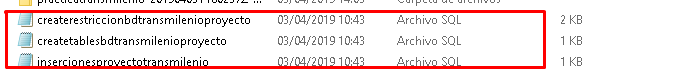
**PRESENTADO POR:** Cesar Andrés Patiño Nieto **CODIGO:** 201822962

Wilmer Ariel Avella Tuta **CODIGO:** 201823020

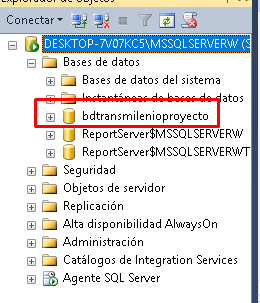
para empezar, es necesario modelar una base de datos donde se pueda evidenciar la infraestructura del Transmilenio para ello utilizamos el powerdesigner para modelar la base de datos

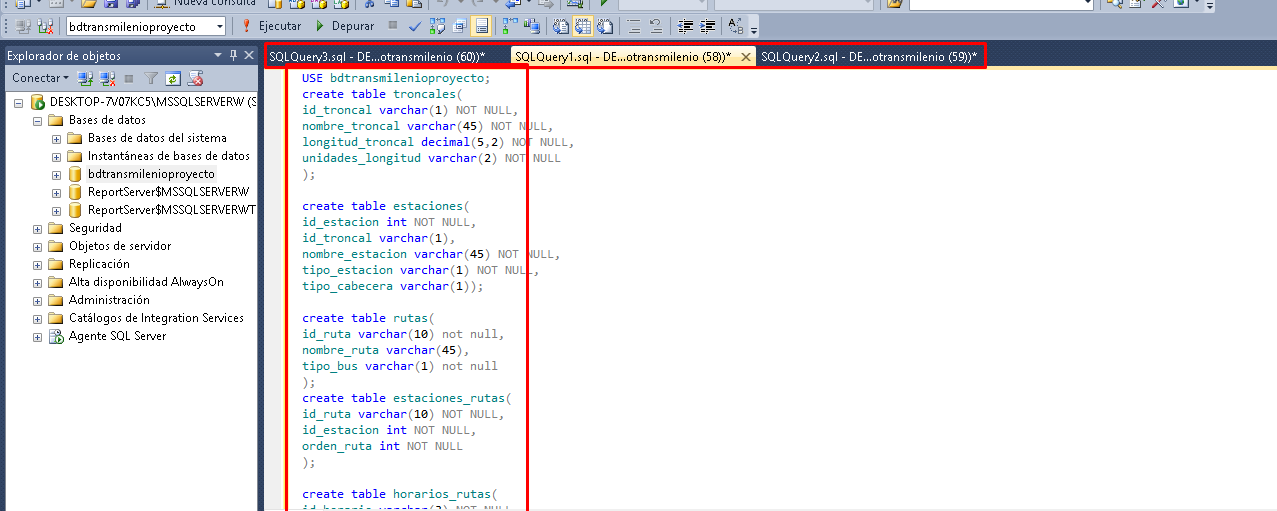


enseguida vamos a desarrollar los diferentes scripts para crear la base de datos del Transmilenio

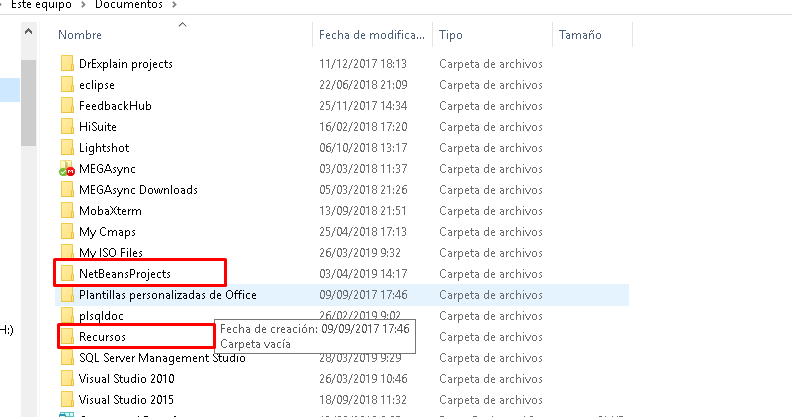


enseguida ya teniendo configurado el super usuario de sql server podemos crear la base de datos para ellos nos situamos en el super usuario creado y nos dirigimos a la carpeta de bases de datos y creamos una base de datos con el nombre de bdtransmilenioproyecto

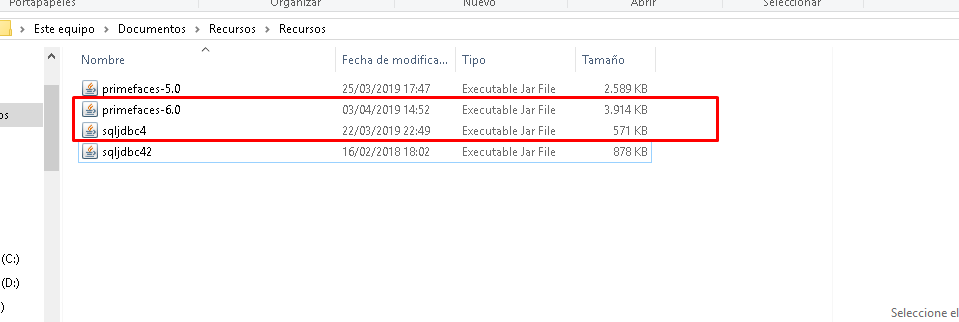


estando en ella damos clic derecho y le damos nueva consulta para copiar y poder ejecutar los diferentes scripts para la creación de la base de datos y ejecutamos 

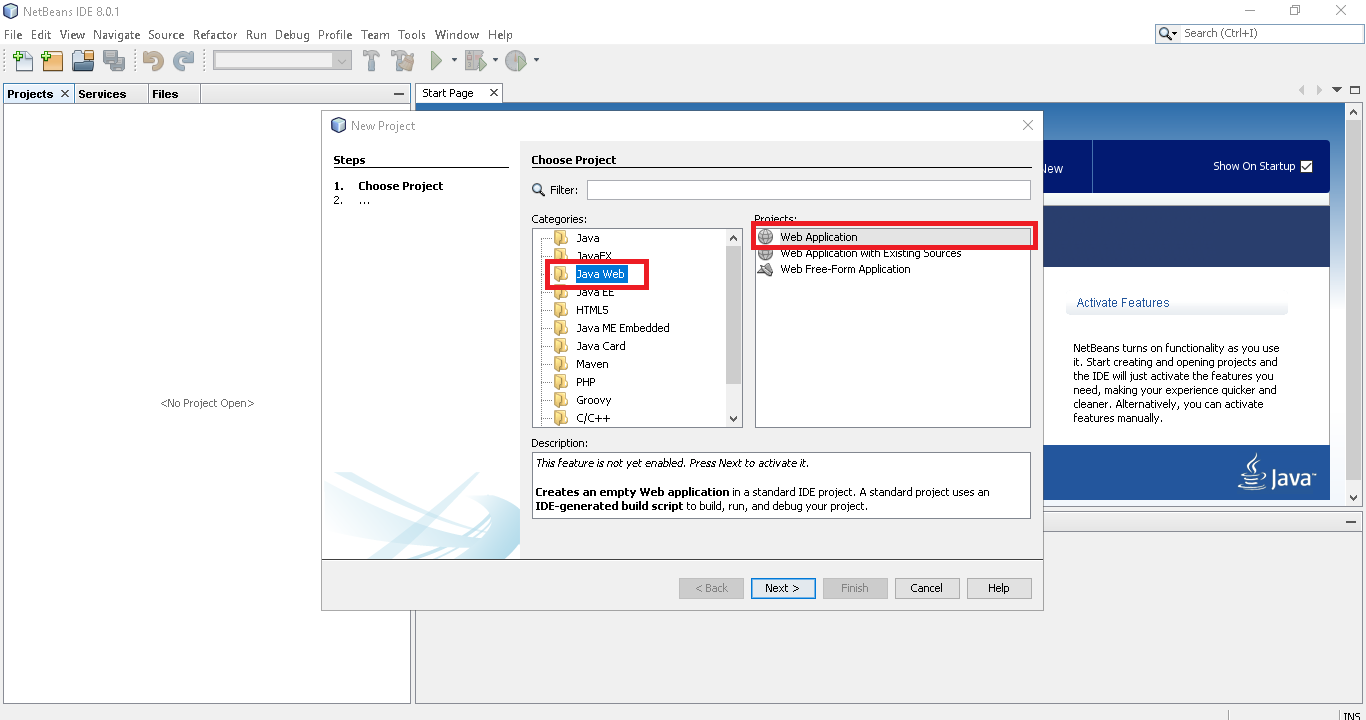
ya teniendo la base de datos configurada y cargada en el sql server vamos a configurar para crear el proyecto, para ello nos dirigimos a donde está la carpeta de NetBeans y en ella creamos una carpeta donde contara con los .jar necesarios para la conexión y el funcionamiento de la aplicación



dentro de la carpeta recursos se encuentran los respectivos .jar

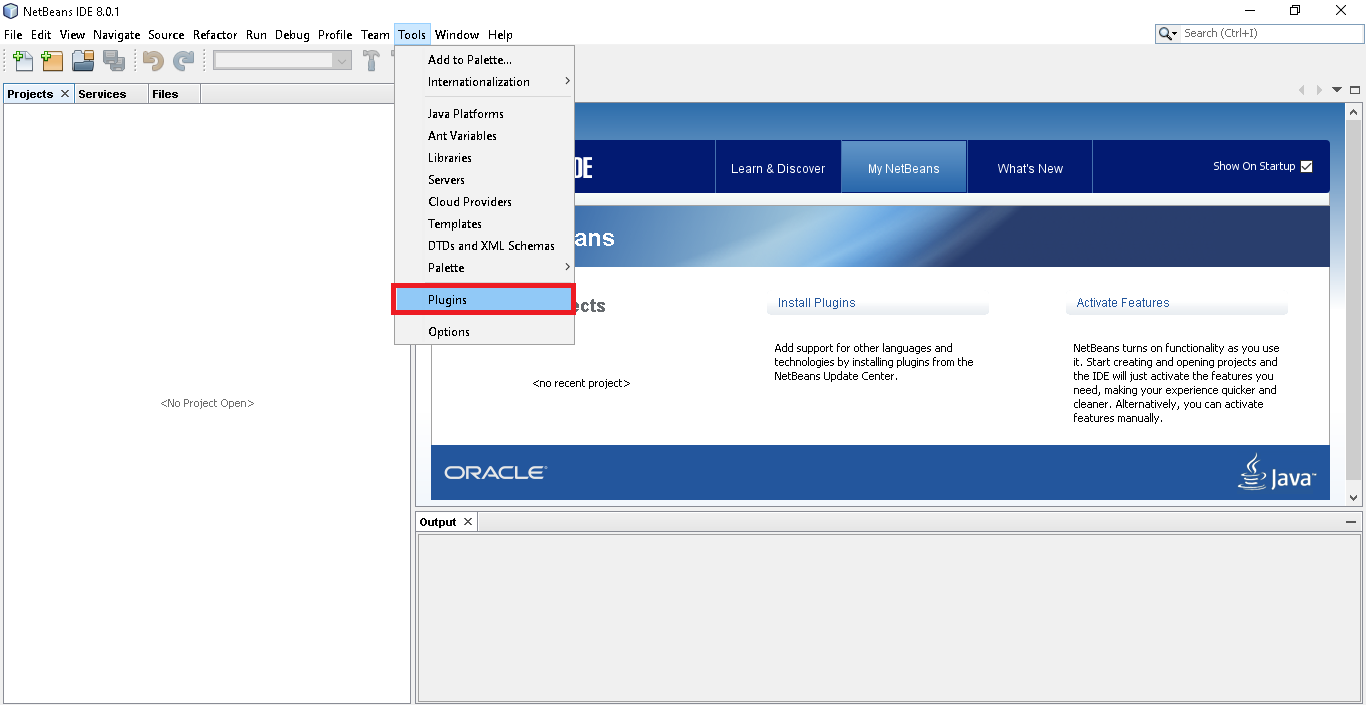


ya teniendo los diferentes componentes necesarios para poder seguir con el desarrollo de la aplicación nos dirigimos a NetBeans para crear el proyecto para ellos verificar si tenemos instalado java web en nuestro entorno de NetBeans

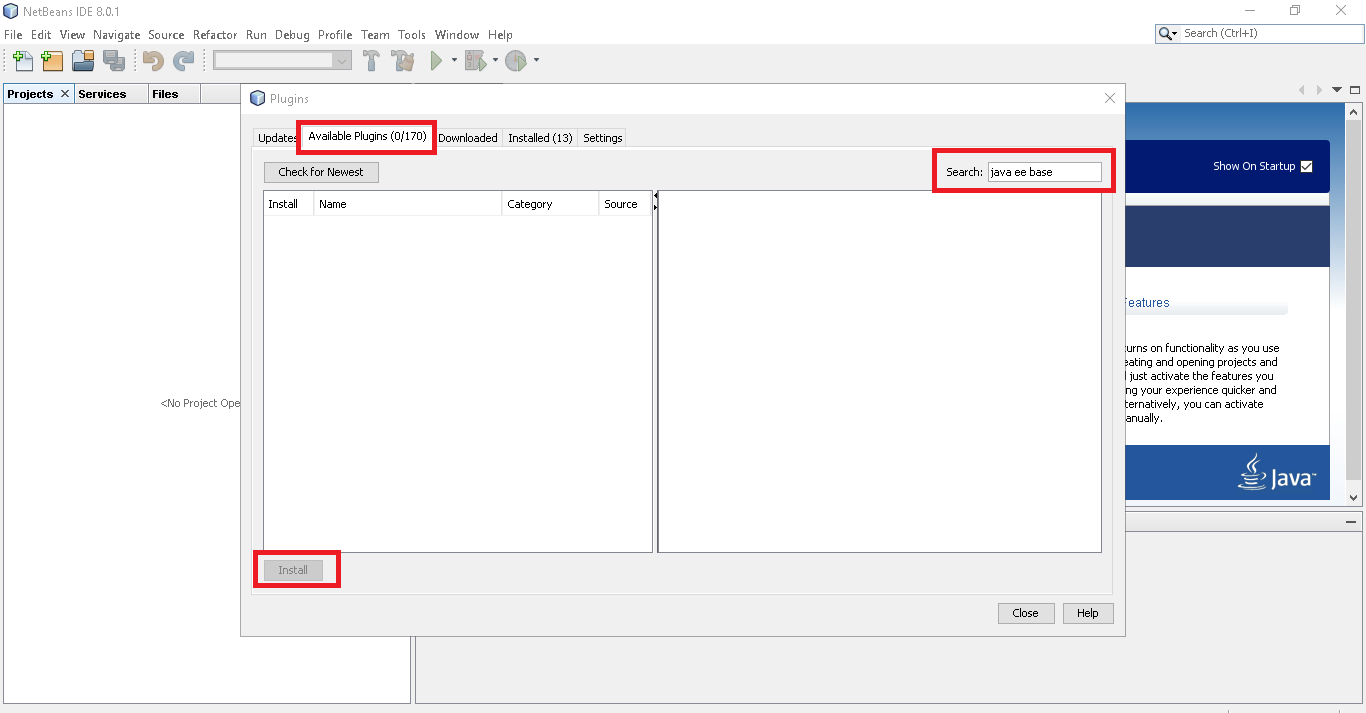


en caso de no tenerlo

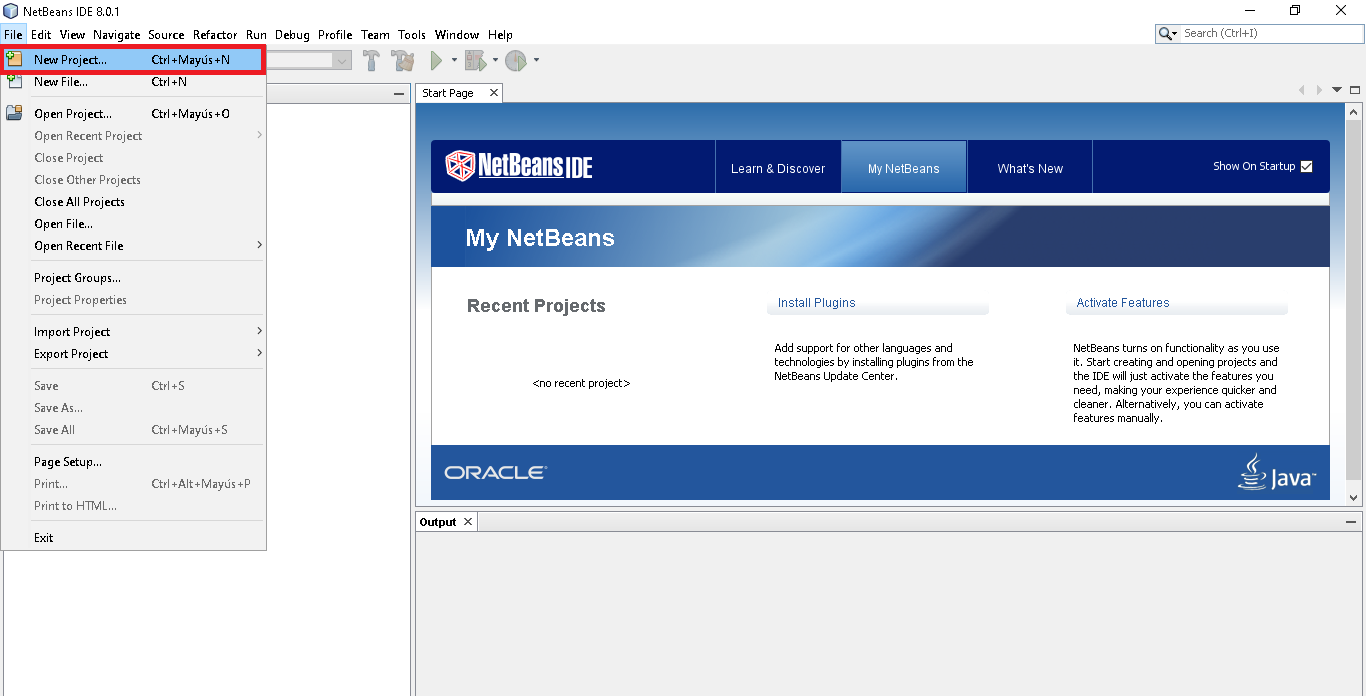
para ello vamos a la pestaña tools y plugin



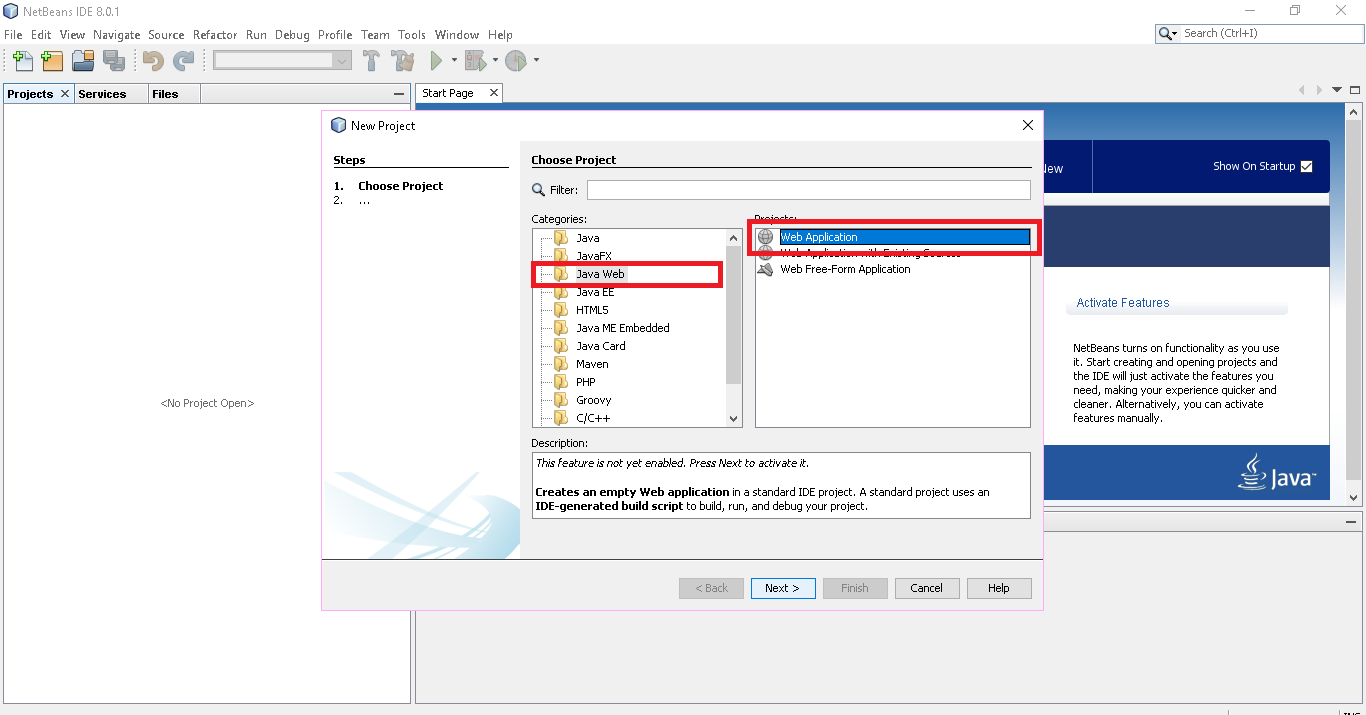
vamos a updates avalaible y escribimos en la parte superior derecha java EE base y si aparece lo seleccionamos y damos install para nuestro caso la version NetBeans 8.1 al instalarlo nos instala los componentes web



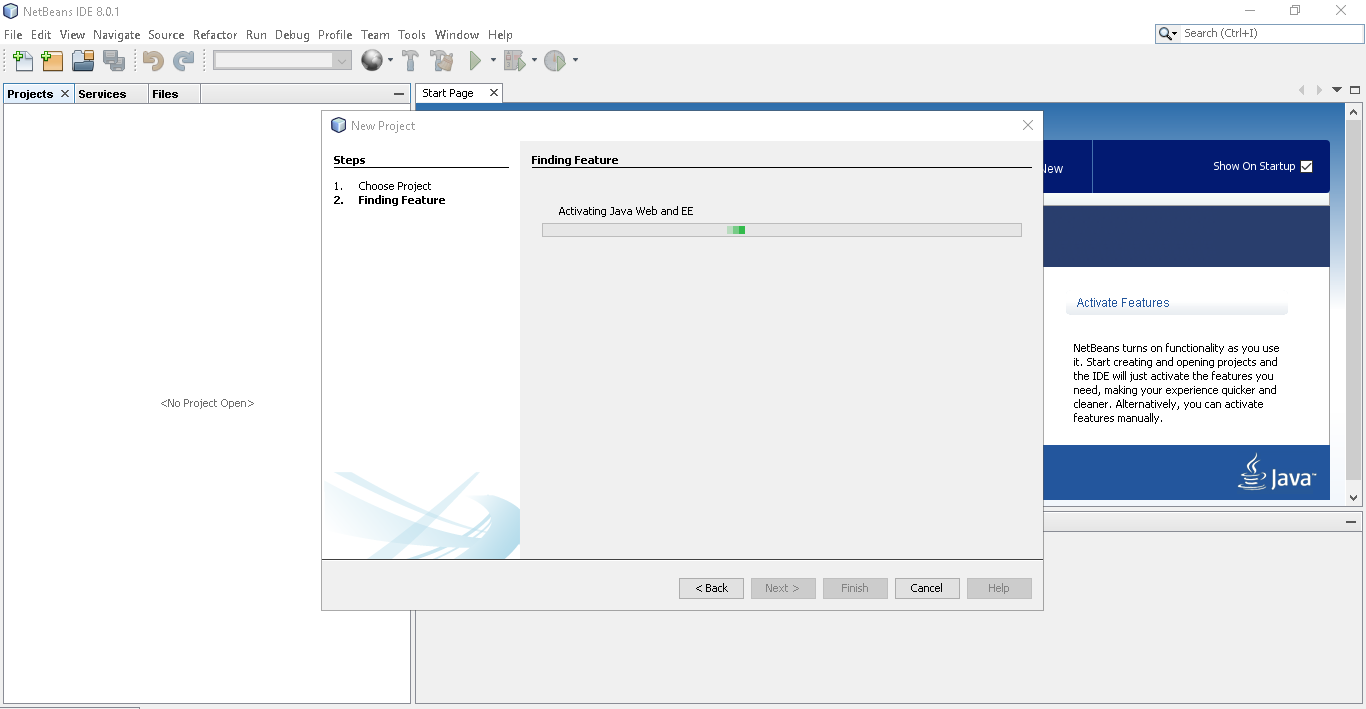
ahora vamos a crear nuestro proyecto vamos a file new project



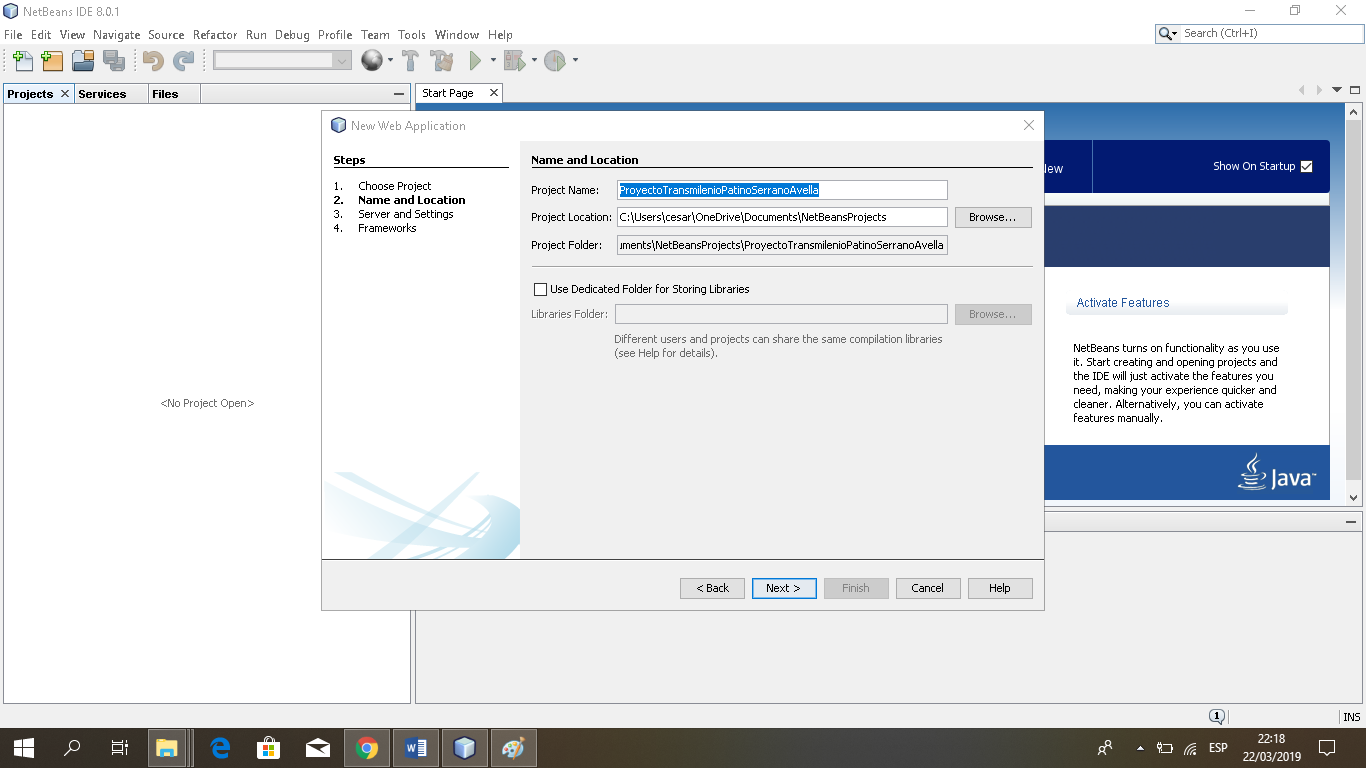
seleccionamos java web y web aplication



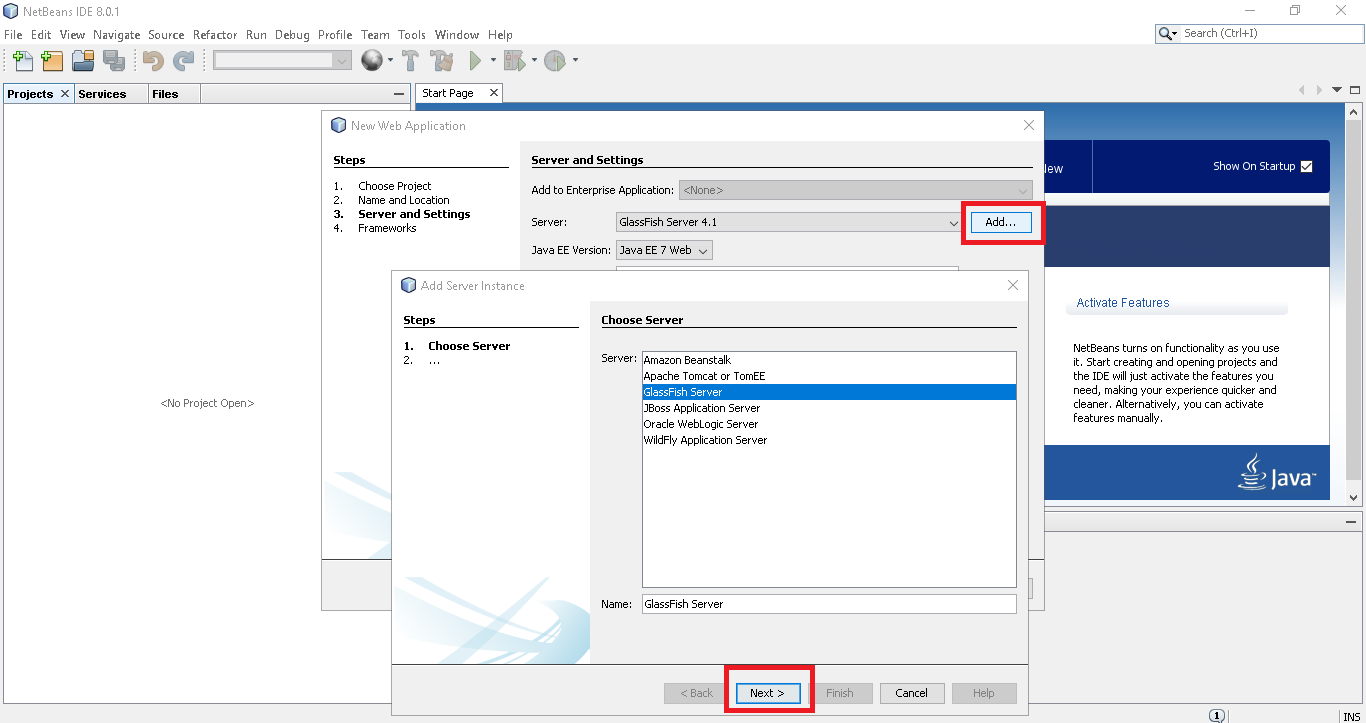
esperamos que le isntale complementos de web



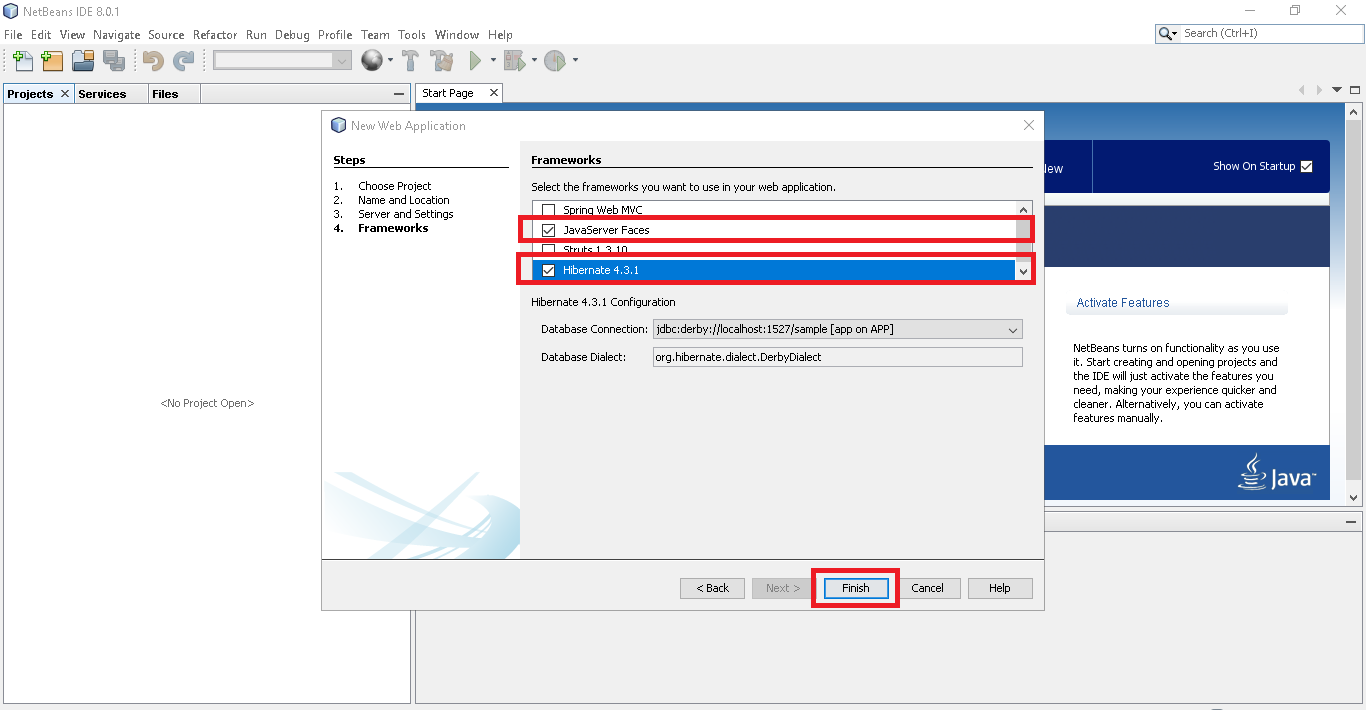
le colocamos el nombre a nuestro proyecto que va a ser proyectotransmileniopatinoavella



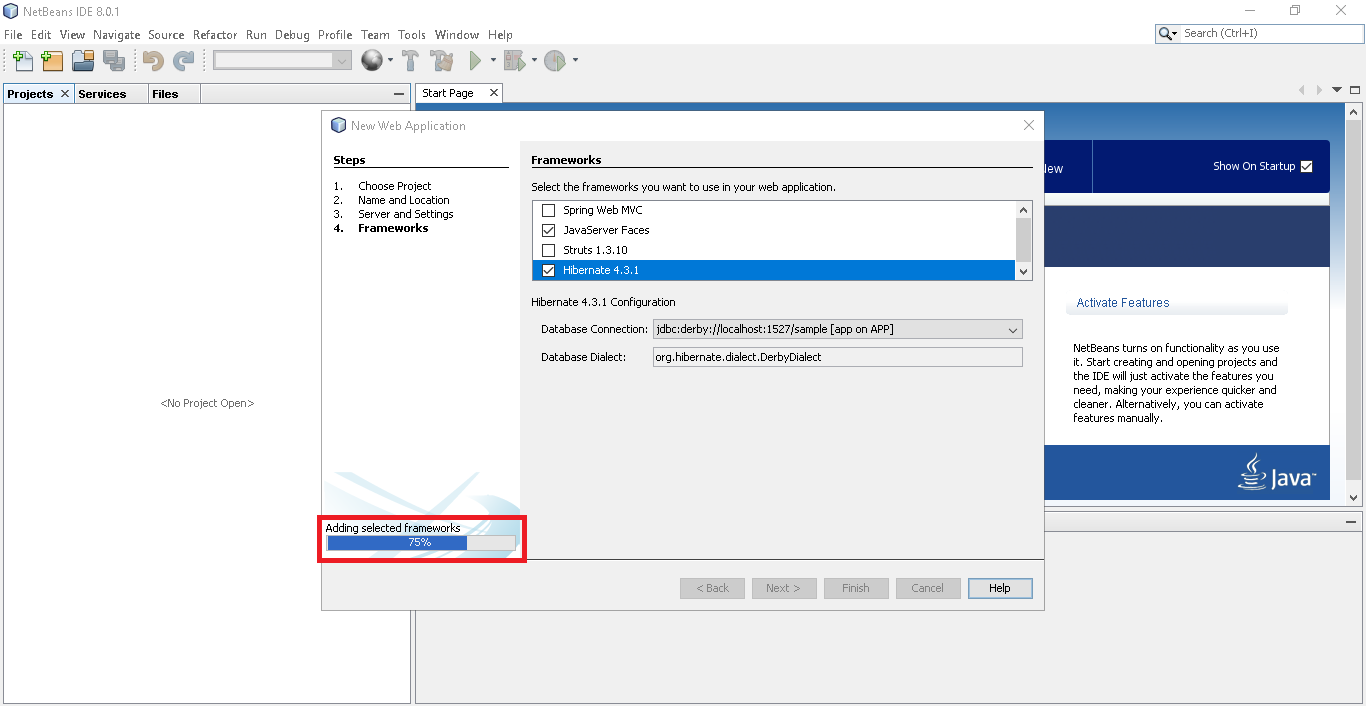
luego es importante ejecutar un servidor de aplicaciones para ello seleccionamos el que nos instala el NetBeans con esto podemos ejecutar las aplicaciones jsp y jsf (es importante porque o si no es imposible ver nuestras aplicaciones hechas en jsf o jsp) que es glashfish existen otros servidores de aplicaciones como tom cat y jboss si queremos instalarlo solo es seleccionarlo darle next, darle el nombre y listo



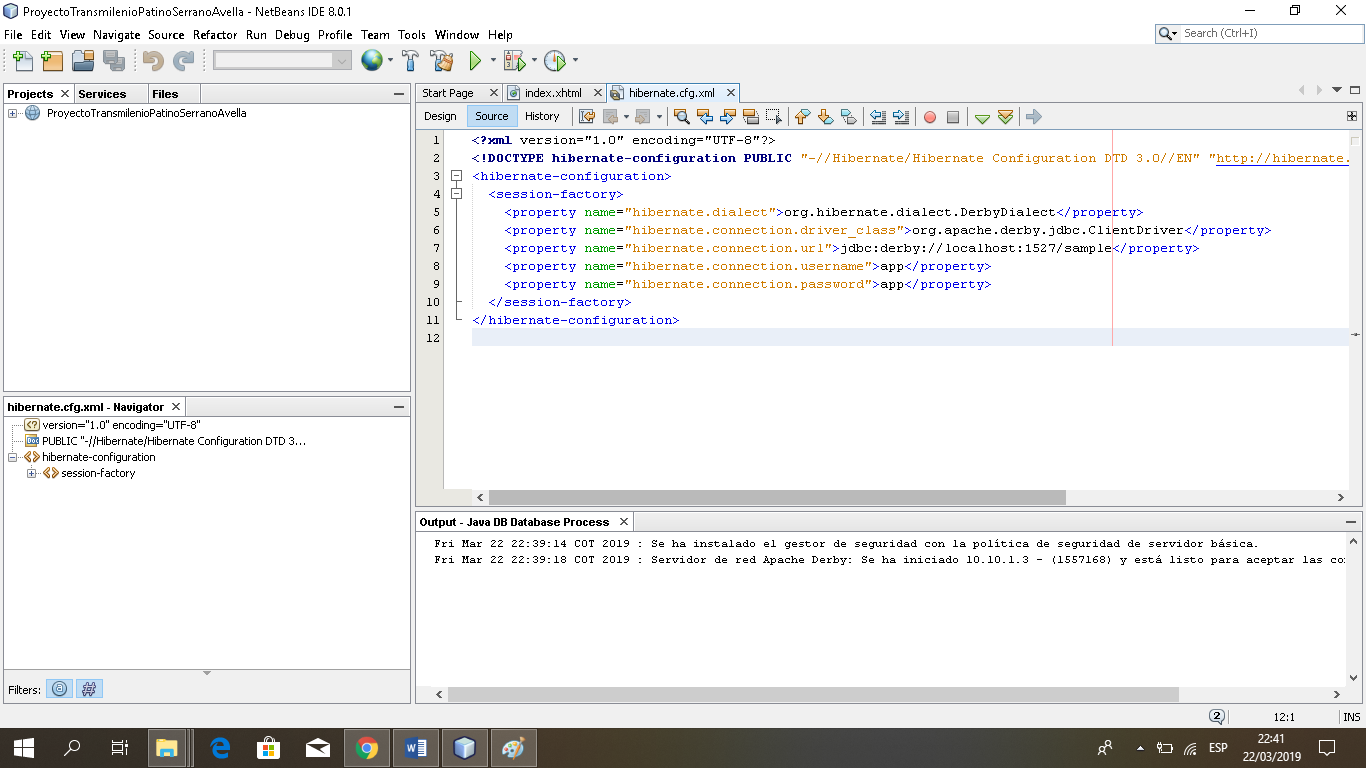
ahora vamos a seleccionar los framework para nuestro proyecto nosotros vamos a seleccionar el framework jsf(java server face) y el framework de persistencia hibernate y le damos finalizar



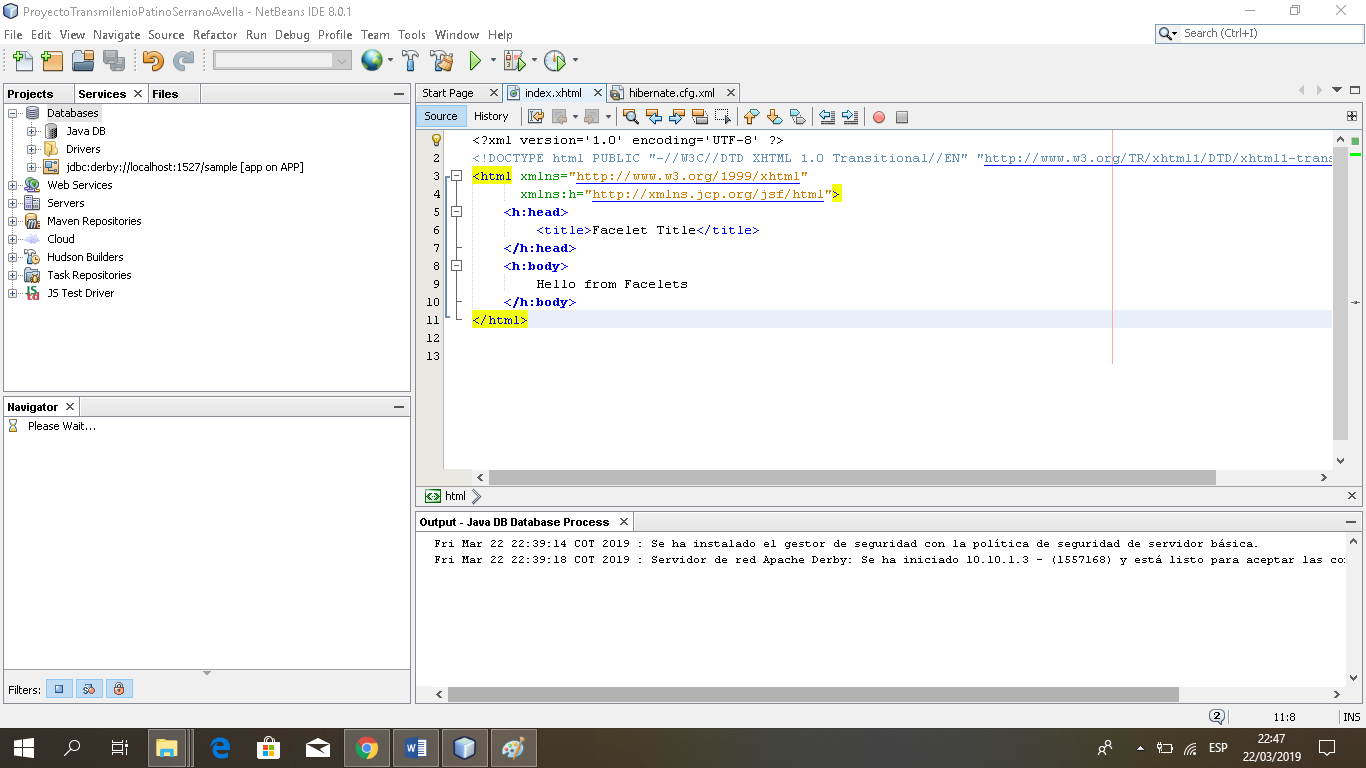
como vemos nos dice que esta isntalando los framework seleccionados



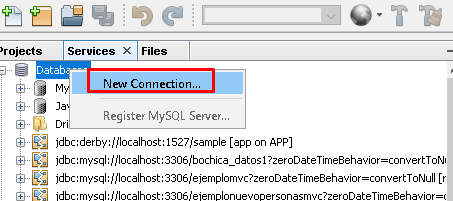
como vemos nos creo un archivo xml que es el de hibernate donde vamos hacer la conexión con la base de datos y la configuración de persistencia con los objetos que vamos a asociar a nuestras entidades de la base de datos

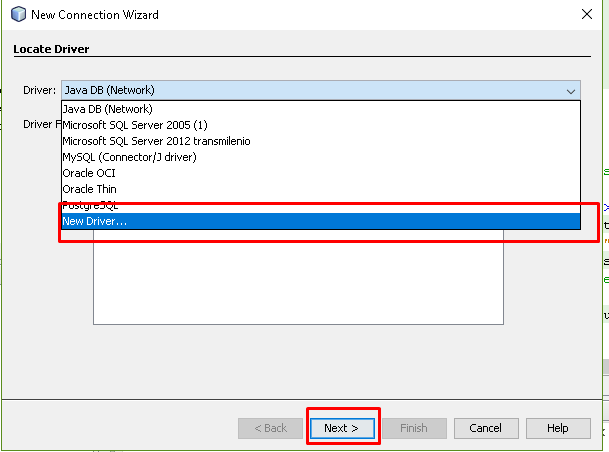


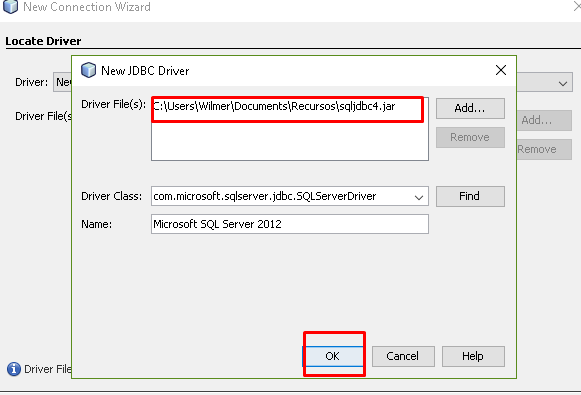
nos creó también el index.xhtml que es el que nos creó al seleccionar jsf



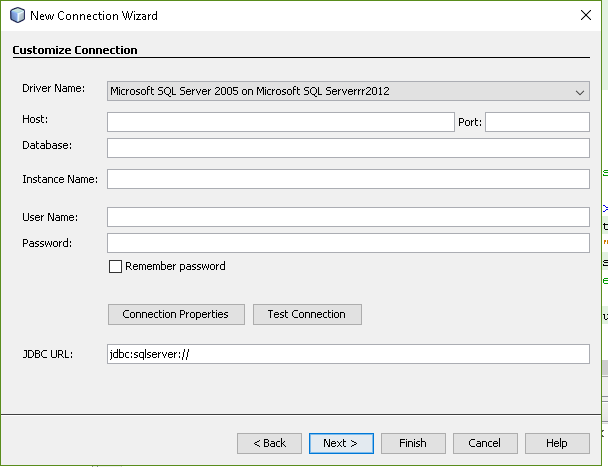
Ya creado el proyecto es necesario que crear una conexión para poder tener conexión con nuestra base de datos que está en sql server para ellos nos dirigimos a services y le damos nueva conexión



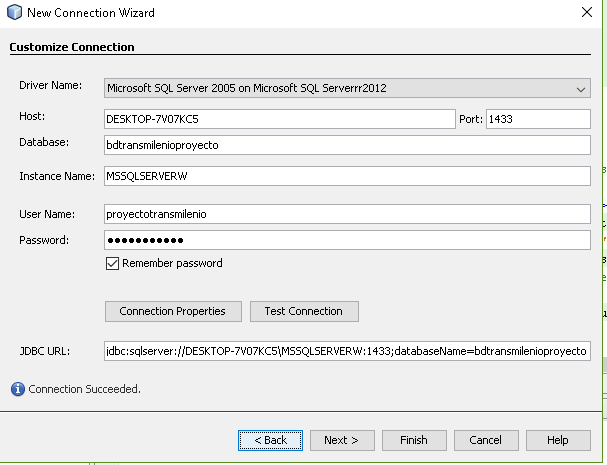
Y agregamos el draip que tenemos en la carpeta recursos de la siguiente manera 



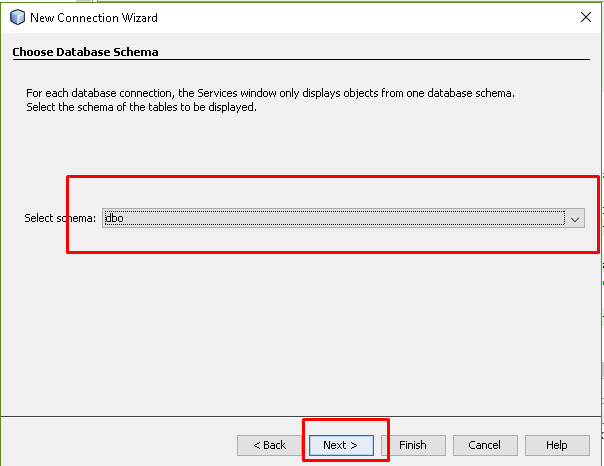
Enseguida nos saldrá la configuración para poder conectarnos a sql server 2012



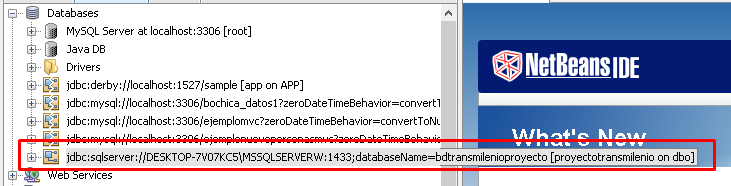
Se completa el formulario correspondiente para seguir con la configuración y verificamos con el tes de conexión



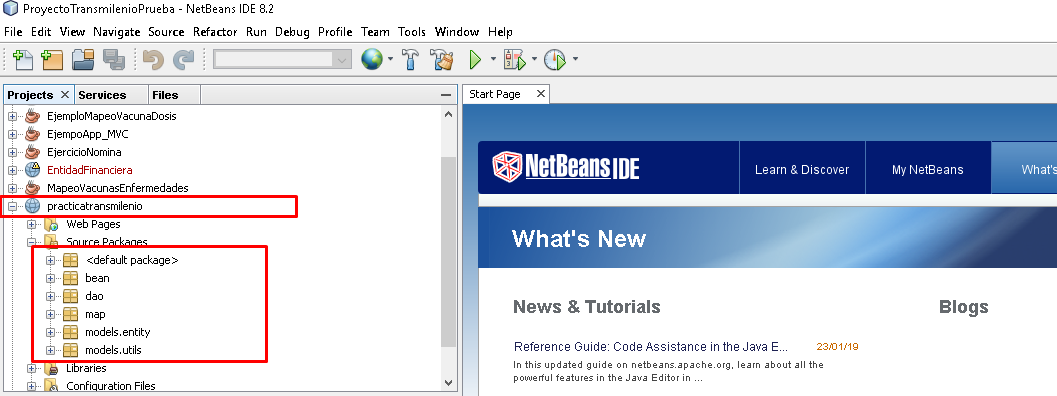
Enseguida seleccionamos la opción de dbo que contienen los mayores privilegios posibles en la base de datos; por lo tanto, tienen privilegios suficientes para crear y administrar la geodatabase.



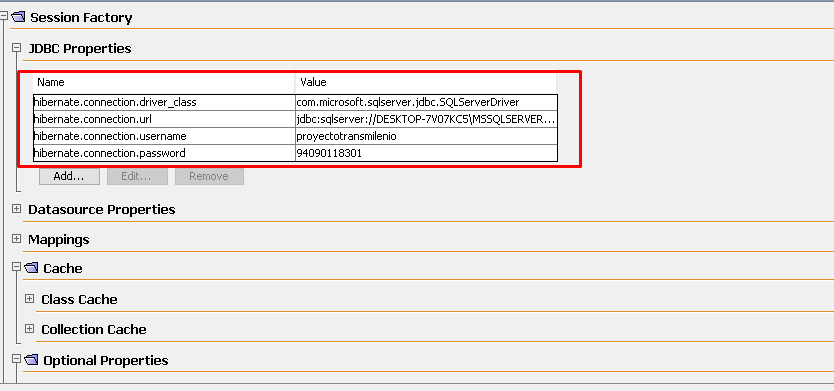
Y le damos finish y automáticamente nos crea la conexión con el motor de bases de datos



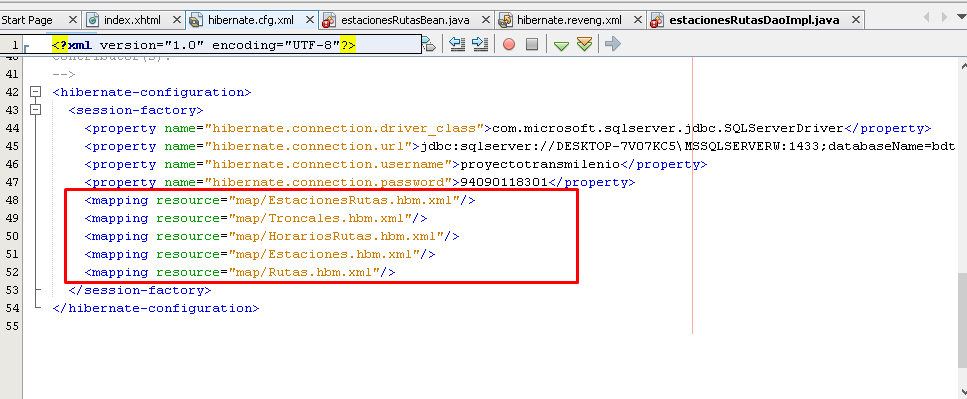
Ya teniendo la configuración de conexión nos dirigimos al proyecto y creamos los diferentes paquetes para el desarrollo de la aplicación



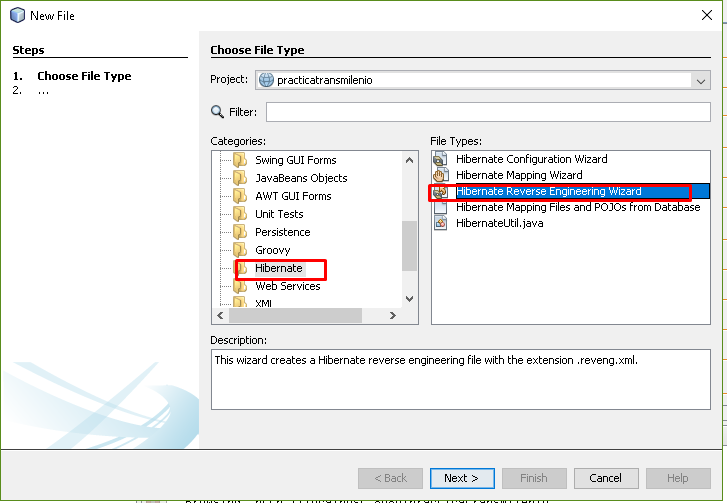
Después de haber creado los paquetes vamos a generar la configuración del hibernate para verificar si nos está haciendo la respectiva conexión con la base de datos y poder desarrollar el proceso de ingeniería inversa y los pojos para las diferentes tablas de nuestra base de datos



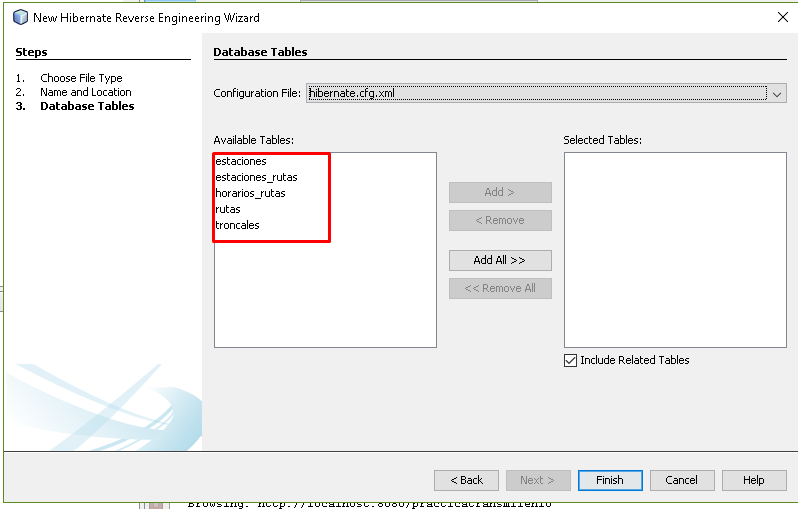
Dentro de esta configuración se puede observar las tablas que queremos mapear

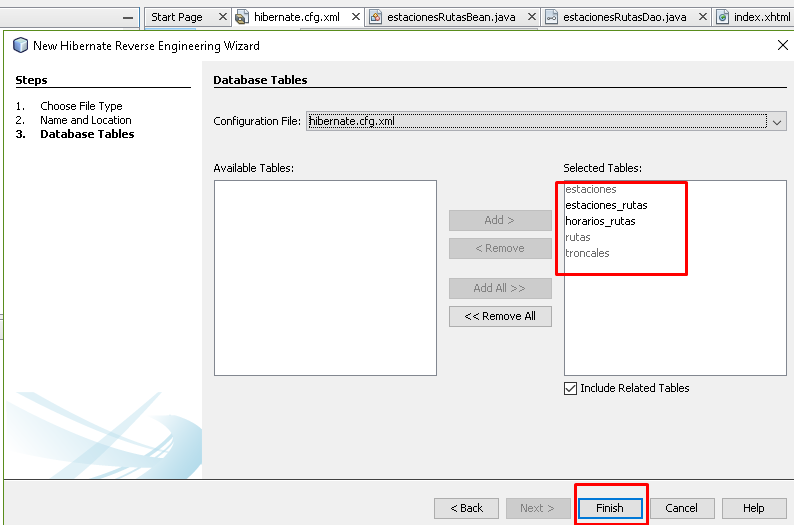


Enseguida vamos a realizar ingeniería inversa para poder mapear posteriormente nuestras tablas de la base de datos y que puedan ser reconocidas por la aplicación para ello nos situamos en el paquete de map

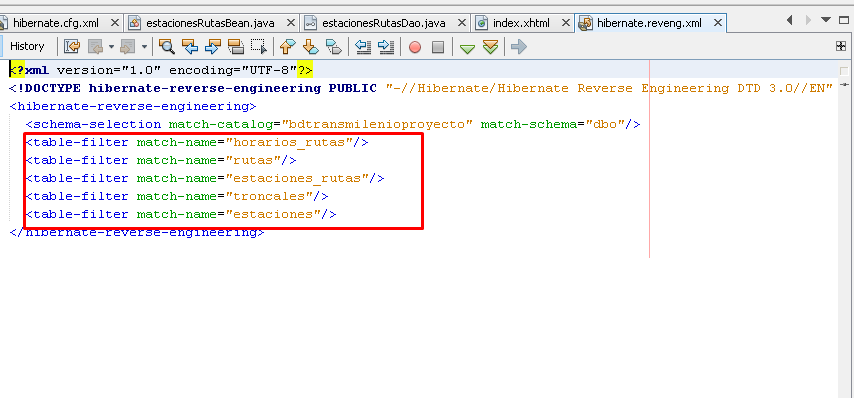


Después de ello nos cargara las diferentes tablas de nuestra base de datos y seleccionamos aquellas tablas a las cuales queremos mapear

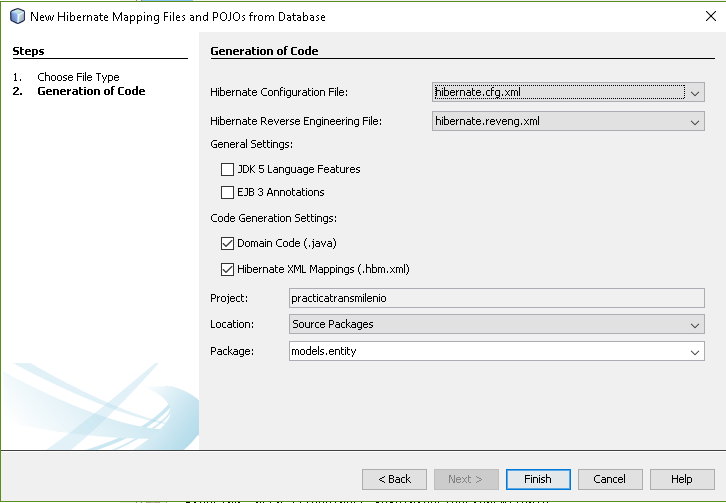




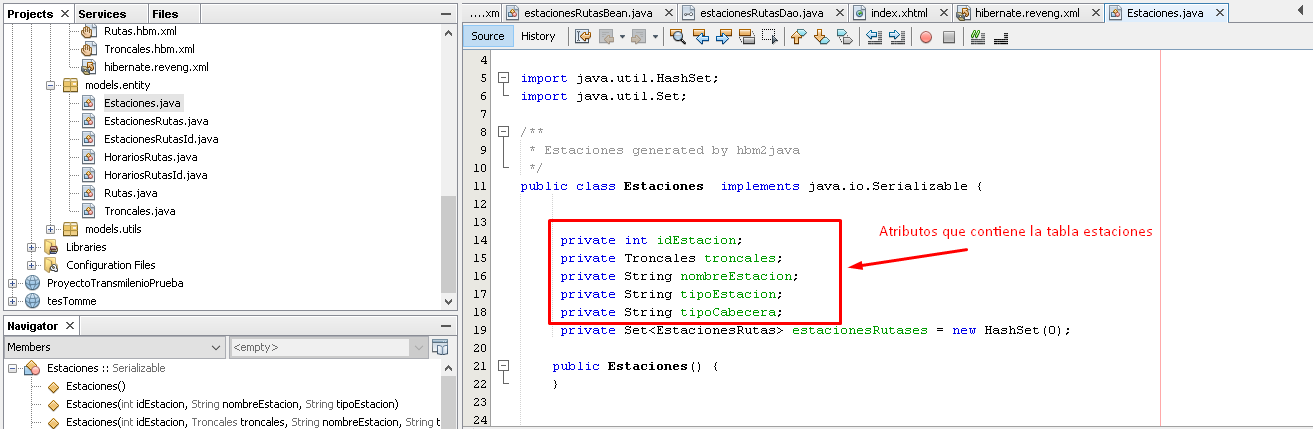
Dentro de esta clase nos arroja una configuración, donde toma el nombre de la base de datos y hace los respectivos filtros necesarios por cada una de las tablas seleccionadas

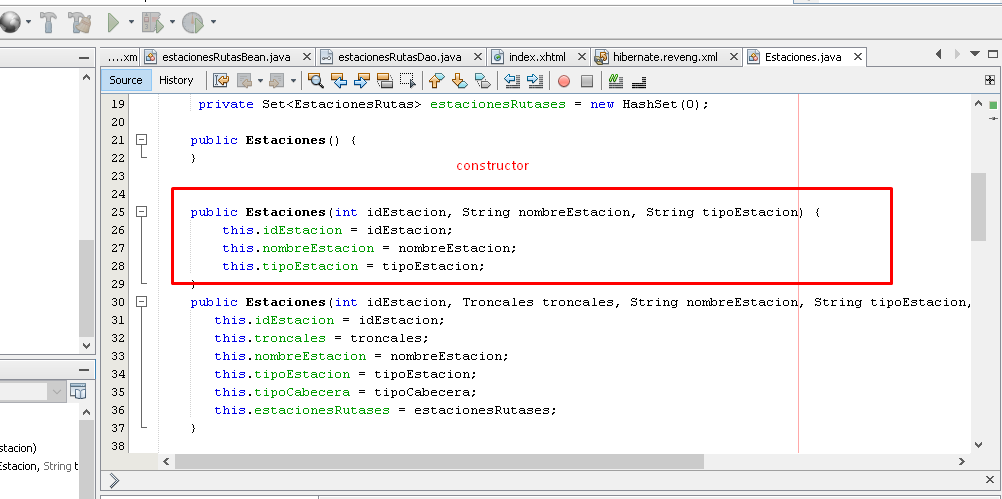


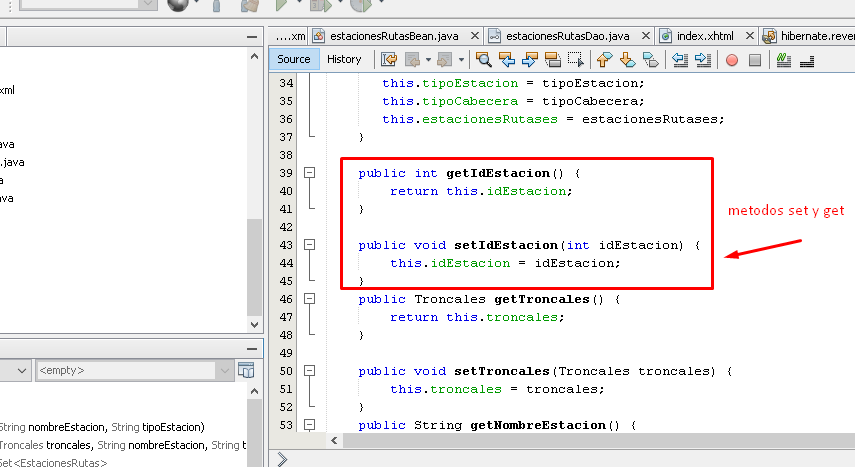
Ya teniendo esto continuamos con el proceso de generar los pojos de las diferentes tablas, donde nos genera unos archivos .hbm y .java

. 

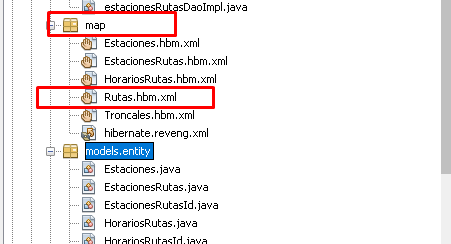
Dentro de las clases .java nos genera los diferentes atributos, constructores y los métodos set y get para poder acceder a los atributos de las tablas.



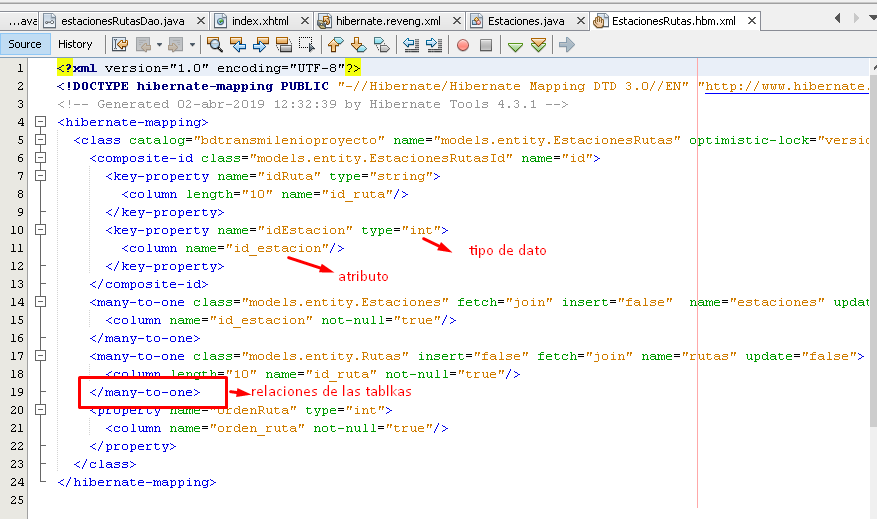




Arrastramos los .hbm al paquete map para tener una mejor distribución

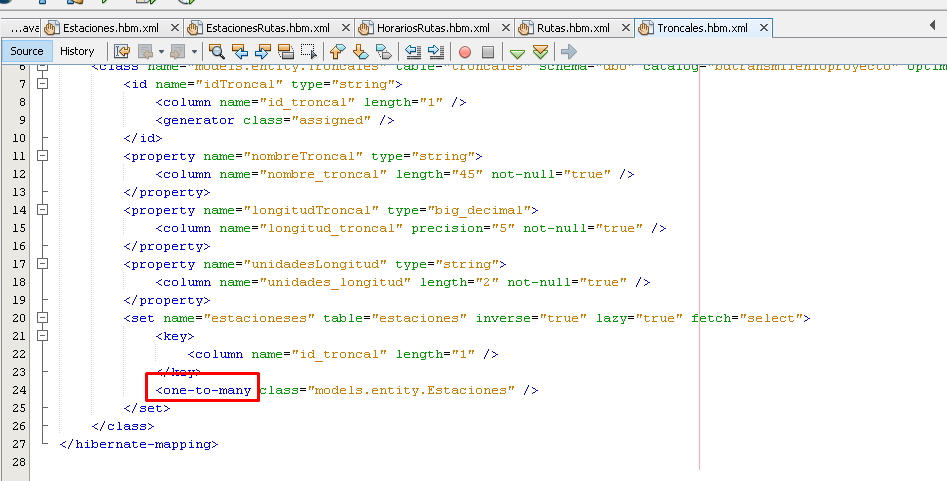


Dentro de estos archivos .hbm contiene los atributos correspondientes de cada tabla y el tipo de dato y de igual forma las relaciones que se generan dentro de cada una de las tablas ya sea de uno a uno, uno a muchos, muchos a uno y de muchos a muchos

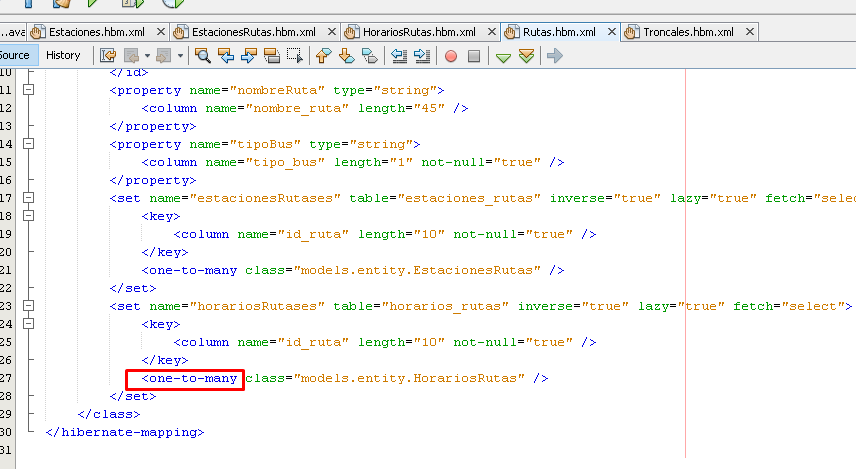


**Relaciones uno a uno:** consiste simplemente en que un objeto tenga una referencia a otro objeto de forma que al persistirse el primer objeto también se persista el segundo, en esta relación va a ser unidireccional es decir que la relación uno a uno va a ser en un único sentido.

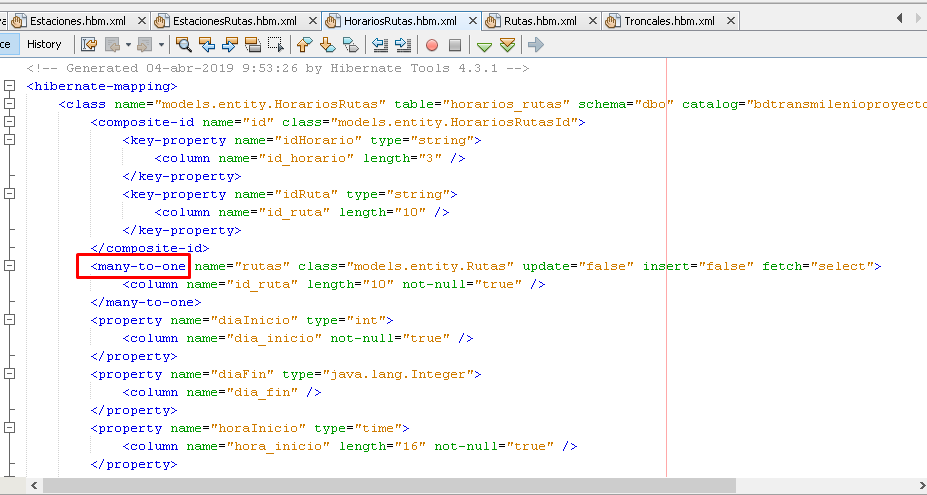
**Relaciones uno a muchos:** consiste simplemente en que un objeto padre tenga una lista sin ordenar de otros objetos hijo de forma que al persistirse el objeto principal también se persista la lista de objetos hijo. Esta relación también suele llamarse maestro-detalle o padre-hijo. Dentro de estas relaciones encontramos la de troncales y rutas



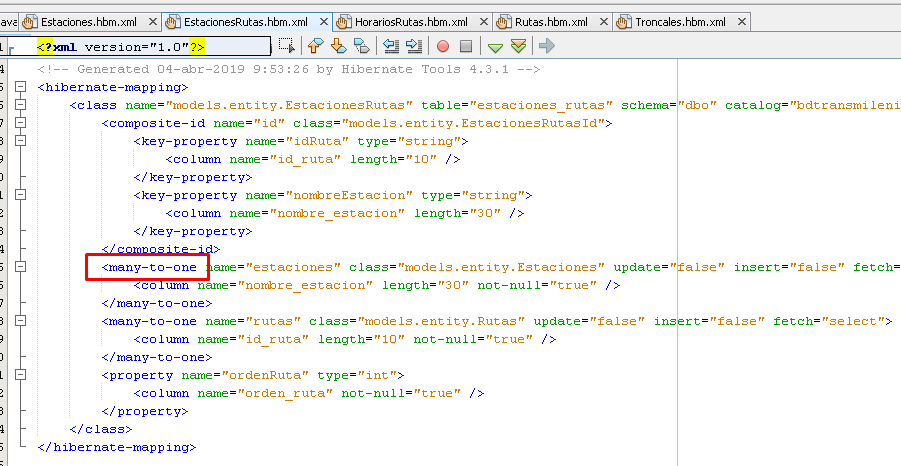
Rutas



**Relaciones de muchos a uno** Estas consiste simplemente en que un objeto padre tenga una lista Ordenar de otros objetos hijo de forma que al persistirse el objeto principal también se persista la lista de objetos hijo. Esta relación también suele llamarse maestro-detalle o padre-hijo. Dentro de estas relaciones encontramos las relaciones que esta entre horarios rutas

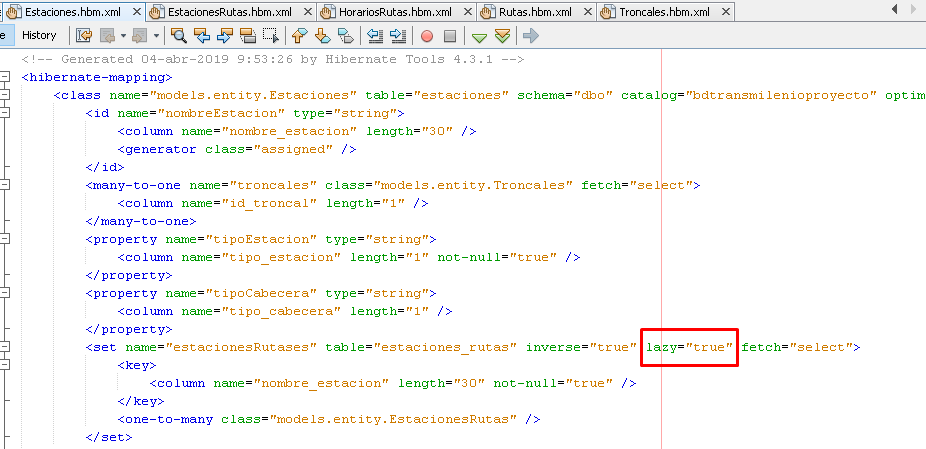


Estaciones rutas



Por último, encontramos **las relaciones muchos a muchos** las cuales consiste en que un objeto A tenga una lista de otros objetos B y también que el objeto B a su vez tenga la lista de objetos A. De forma que al persistirse cualquier objeto también se persista la lista de objetos que posee.

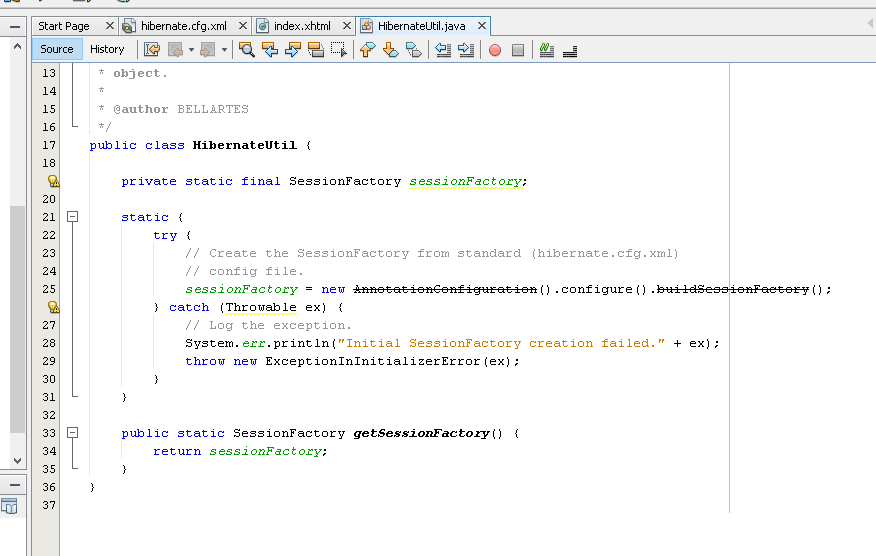
Dentro de los archivos .hbm encontramos los parámetros lazy los cuales Al especificar una relación en un mapeo de Hibernate especificar el “cuando”, por defecto lazy es igual a true esto significa que la colección no se recupera de la base de datos hasta que se hace alguna operación sobre ella.



Por otro lado, encontramos los atributos fetch estos definen que SQLs se van a lanzar para recuperar la información

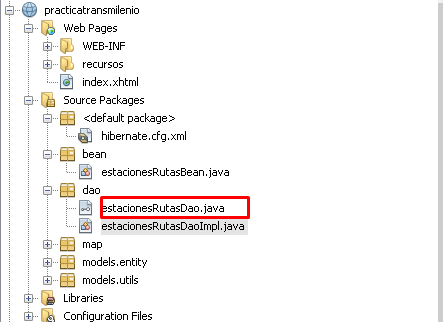


Enseguida dentro del paquete models.utils generamos el archivo hibernate.utils el cual es un archivo de comunicación entre hibernate y la base de datos y las respectivas excepciones.

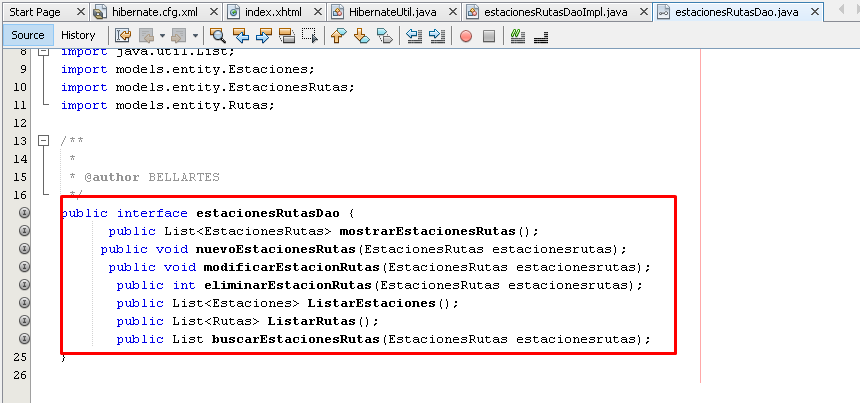


**MOSTRAR GESTION ESTACIONES RUTAS**

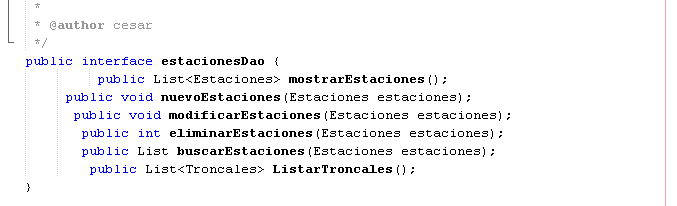
Ahora dentro e paquete dao vamos a generar una nueva clase donde vamos a generar un método donde se pueda mostrar en la página índex.



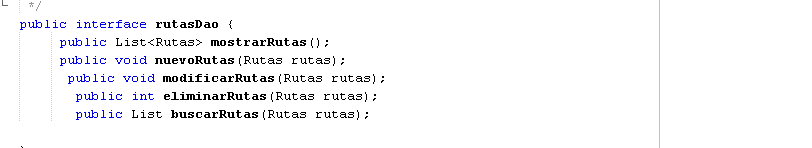
Dentro de esta clase vamos a generar los respectivos métodos para llenar, mostrar, eliminar y modificar una lista de estaciones rutas el cual le generamos el siguiente código, cabe resaltar que en esta clase se hace el llamado a la entidad estacionesrutas la cual esta en el paquete models.entity



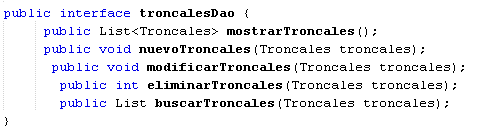
De igual forma para estaciones



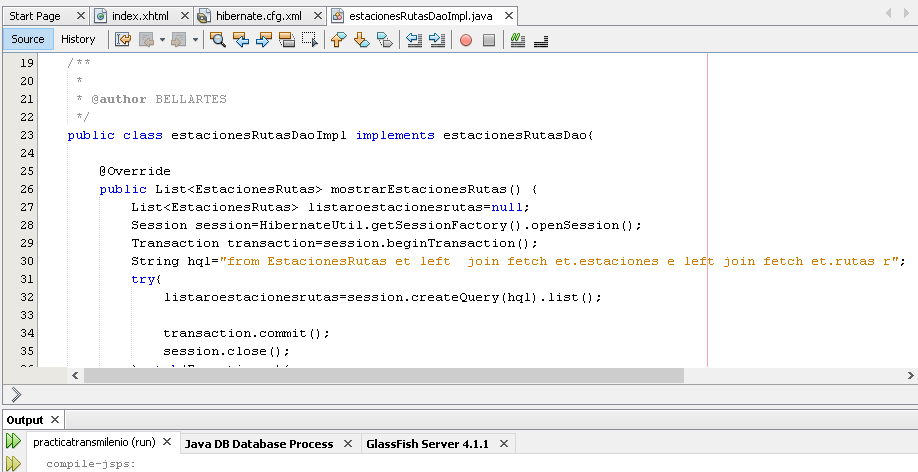
También para rutas



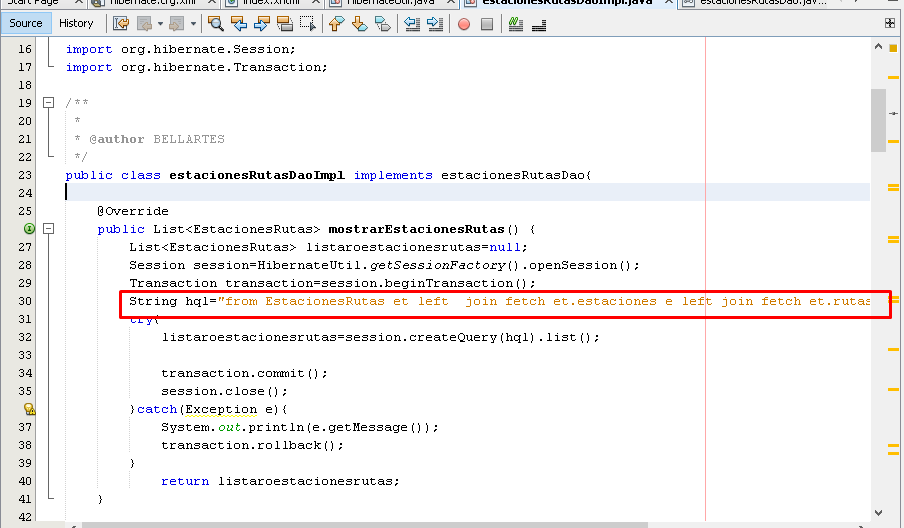
y por ultimo para las troncales



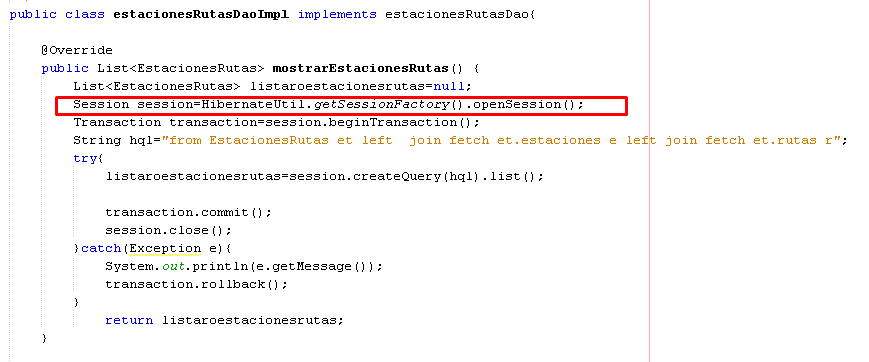
De igual forma creamos las clase normalita de java esto para que implemente los métodos creados en la clases anterior mente descritos

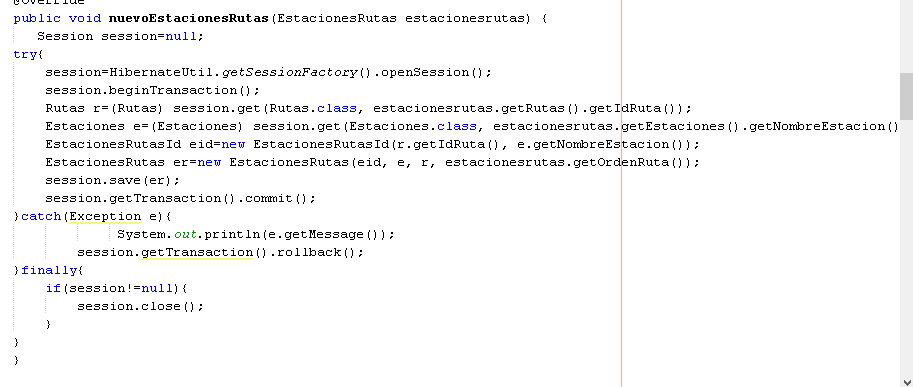


Para generar el método de mostrar estacionesrutas es necesario generar el siguiente código, dentro de este método se crea una variable hql donde se hace una consulta hql ya que por utilizar el lenguaje hibernate la consulta es diferente a las consultas normalitas de sql

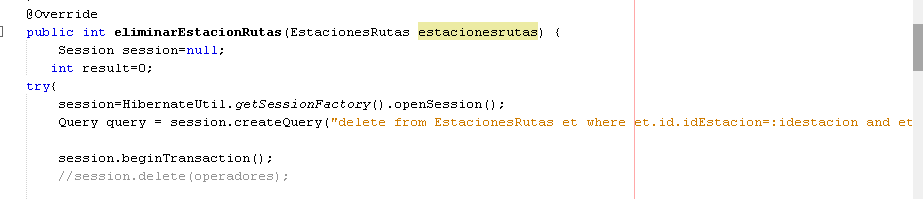


De igual forma dentro del método insertar podemos ver que se crea una session con nuestra base de datos para ello utilizamos el hibernate entro del método insertar podemos ver que se crea una session con nuestra base de datos para ello utilizamos el hibernate para nuestro caso estamos utilizando un pojo que tiene las llaves primarias de estaciones y rutas por ello creamos instancias de los pojos rutas y estaciones donde se verifico que los datos que estamos ingresando existen en los registros para poder modificarlos luego mediante la session hacemos la respectiva modificaciones

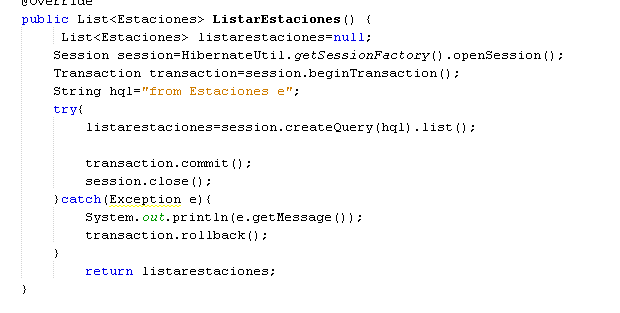
Enseguida vamos a realizar el método de nueva estación ruta la cual podrá registrar una nueva inserción a la base de datos, para ello debemos traer los atributos necesarios para llevar esta inserción, este lo atrapamos con los métodos get que están dentro de los pojos estaciones rutas.



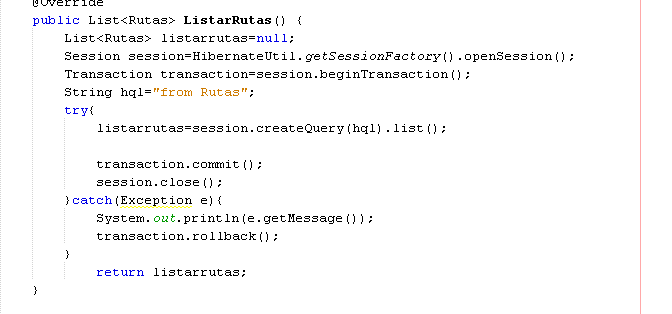
Después creamos el método que se pueda eliminar un registro de las estaciones rutas y a la ves se actualice dentro de la base de datos



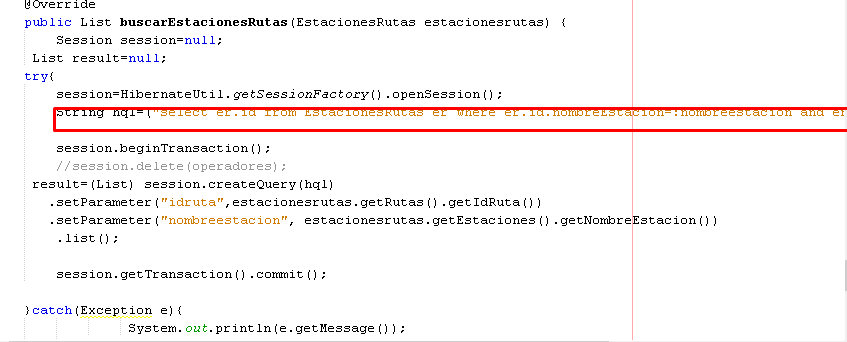
Enseguida creamos el método de listar estaciones en donde se crea una sesión que atrape los atributos de los pojos correspondientes de la clase estaciones, y dentro de este método también es necesario implementar una consulta hql para que nos muestre todas las estaciones



Enseguida dentro de esta misma clase generamos una lista para que nos atrape todas las rutas que están insertadas en la base de datos y nos la muestre, para ello es necesario implementar una consulta hql para que nos traiga todos los datos de esta tabla.

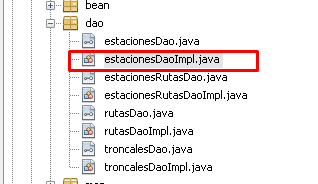


Por ultimo creamos el método de buscar estaciones rutas para ello es necesario crear una sesión en donde nos traiga los atributos correspondientes a la tabla estacionesrutas esto se hace a través de los métodos get, de igual forma se debe generar una consulta para que nos traiga los datos tanto de las estaciones como de las rutas

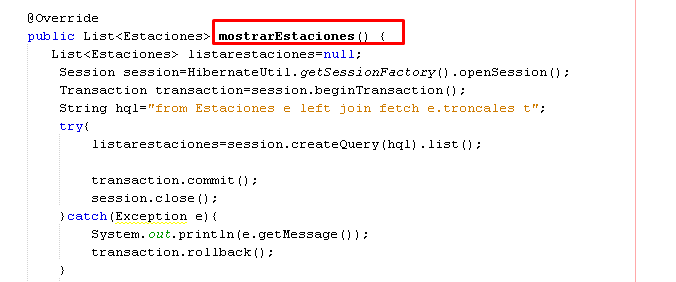


**MOSTRAR GESTION ESTACIONES**

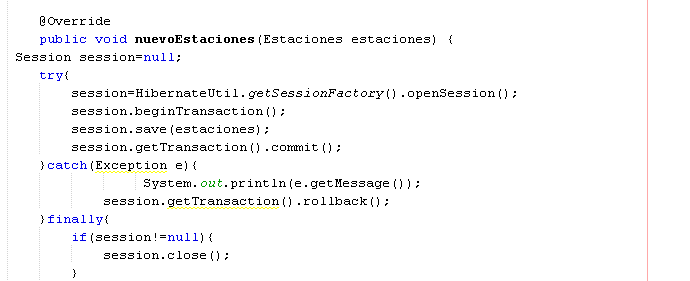
Dentro del paquete dao creamos una nueva clase llamada estaciones.dao para generar los métodos de listar, mostrar, buscar, modificar y eliminar



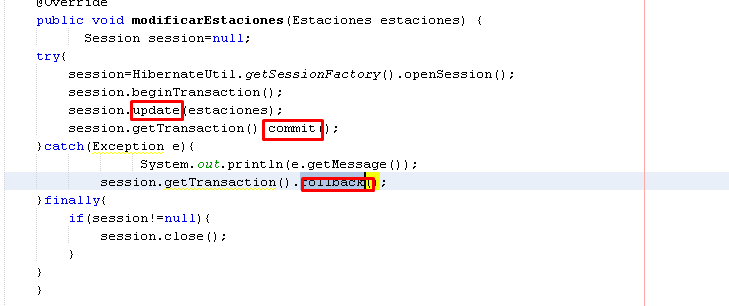
Enseguida vamos a desarrollar el método de mostrar estaciones en donde es necesario implementar un método llamado List para me muestre la lista de las estaciones y crear una sesión para que me traiga los métodos correspondientes a los atributos de esta clase que se encuentran en los pojos creados anteriormente, enseguida creamos una consulta hql para que me traiga todas las estaciones



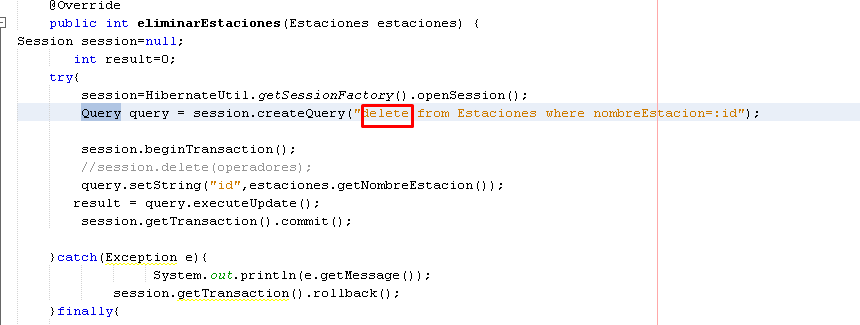
Enseguida configuramos el método de nueva estación para ello creamos una sesión para que me atrape los nuevos registros dentro de la tabla estaciones



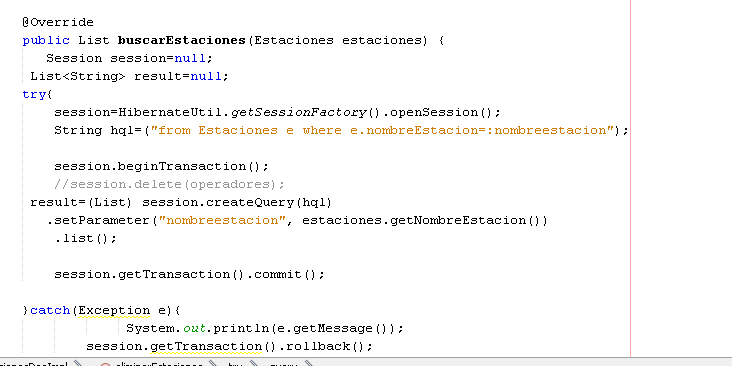
Ahora creamos el método de modificar estación la cual le colocamos un parámetro que atrape todas las estaciones insertadas en la base de datos, enseguida creamos una sesión la cual va modificar dichos datos utilizando el parámetro update, después de ello guardamos los cambios correspondientes con un commit, en caso que no quiera restablecer a las inserciones anteriores utilizamos el parámetro rollback



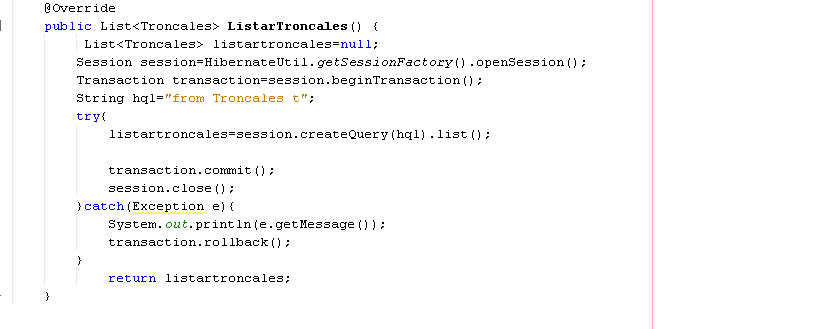
Dentro de esta misma clase generamos el método de eliminar estación la cual es necesario generar un Query para hacer una consulta la cual tendrá el parámetro de delete y el parámetro por el cual queremos borrar es por el nombre de la estación que sea igual al id y esta al hacer eliminado se actualice dentro de la base de datos.



Enseguida creamos el método de buscar en donde en el se crea una consulta hql donde nos mostrara los nombres de todas las estaciones que están insertadas hasta el momento en la base de datos y como resultado nos traerá la lista por nombre de la estación.

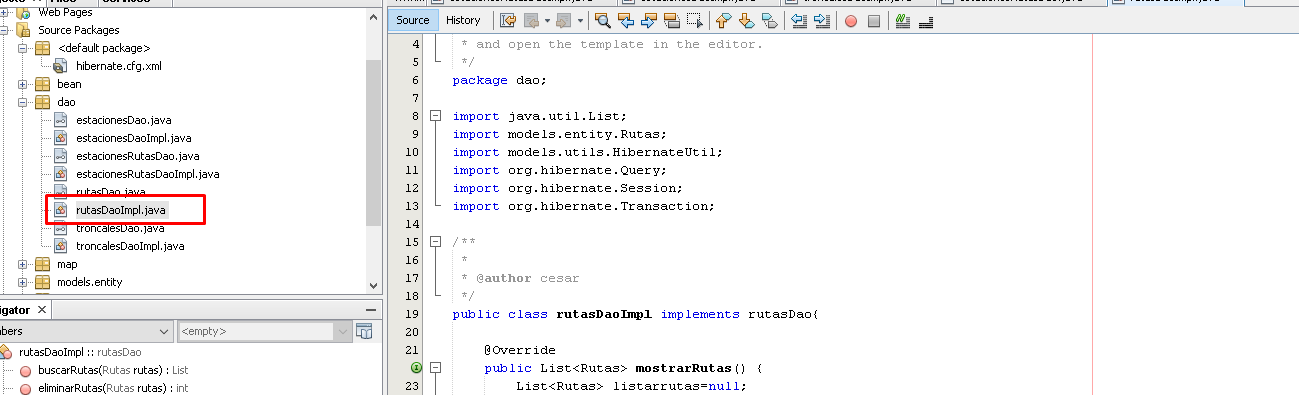


Y por último creamos el método de listar troncales en donde se crea una sesión que atrape los atributos de los pojos correspondientes de la clase troncales, dentro de este método también es necesario implementar una consulta hql para que nos muestre todas las troncales

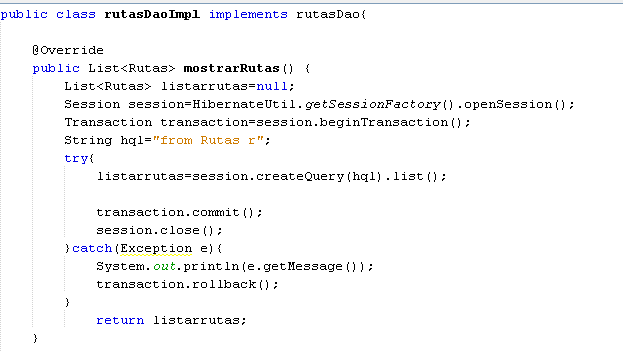


**MOSTRAR GESTION RUTAS**

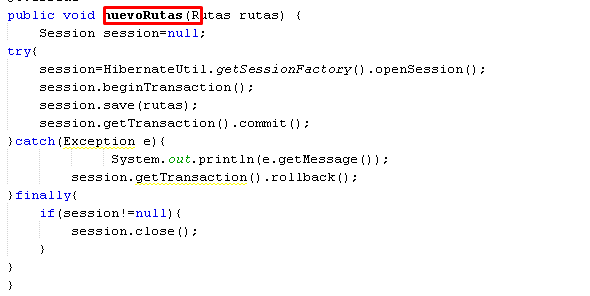
Creamos una nueva clase la cual llamaremos rutasdaoimplement para generar los diferentes métodos de mostrar rutas, nueva ruta, modificar ruta, eliminar ruta, y buscar rutas



Enseguida vamos a desarrollar el método de mostrar rutas en donde es necesario implementar un método llamado List para que me muestre la lista de las rutas y crear una sesión para que me traiga los métodos correspondientes a los atributos de esta clase que se encuentran en los pojos creados anteriormente, enseguida creamos una consulta hql para que me traiga todas las rutas



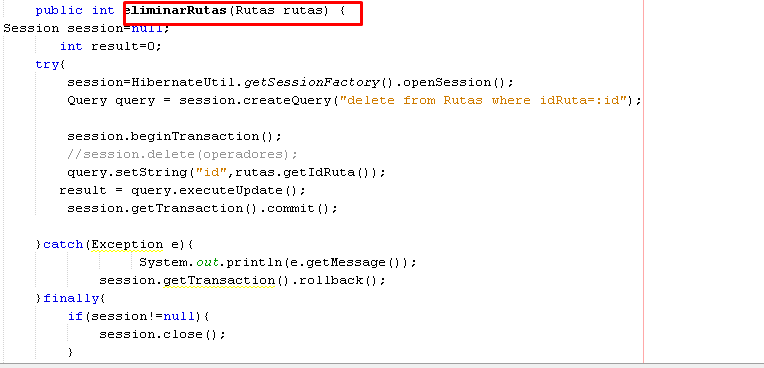
Enseguida configuramos el método de nueva ruta el cual cuenta con un parámetro de rutas para ello creamos una sesión para que me atrape los nuevos registros dentro de la tabla rutas



Ahora creamos el método de modificar rutas la cual le colocamos un parámetro que atrape todas las rutas insertadas en la base de datos, enseguida creamos una sesión la cual va modificar dichos datos utilizando el parámetro update, después de ello guardamos los cambios correspondientes con un commit, en caso que no quiera restablecer a las inserciones anteriores utilizamos el parámetro rollback



Dentro de esta misma clase generamos el método de eliminar rutas la cual es necesario generar un Query para hacer una consulta la cual tendrá el parámetro de delete y el parámetro por el cual queremos borrar en este caso es por el id de la ruta, esta al haber sido eliminado debe actualizar los registros dentro de la base de datos

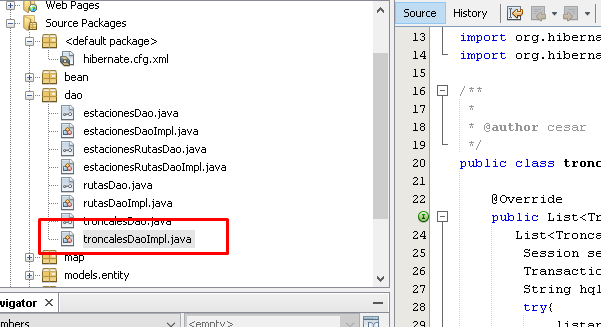
.

Enseguida creamos el método de buscar en donde en él se crea una consulta hql donde nos mostrara los id de las rutas que están insertadas hasta el momento en la base de datos y como resultado nos arrojara dentro de la consulta los id.

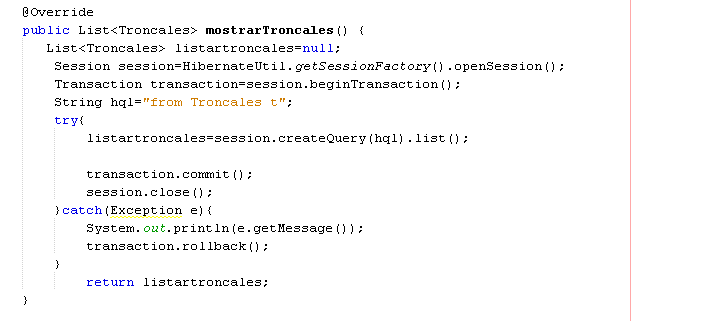


**MOSTRAR GESTION TRONCALES**

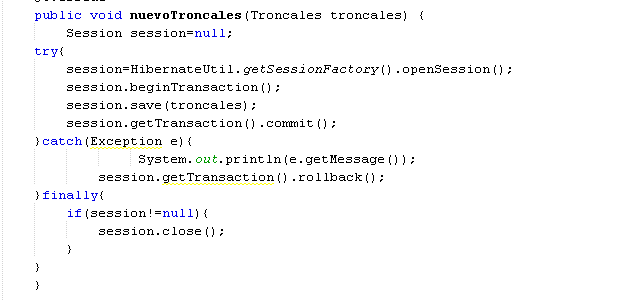
Creamos una nueva clase la cual llamaremos troncalesdaoimplement para generar los diferentes métodos de mostrar tronca , nueva troncal, modificar troncal, eliminar troncal, y buscar troncal



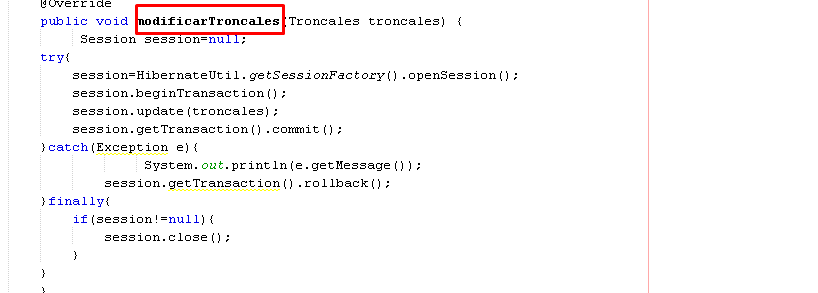
Enseguida vamos a desarrollar el método de mostrar troncal en donde es necesario implementar un método llamado List para que me muestre la lista de las troncales y crear una sesión para que me traiga los métodos correspondientes a los atributos de esta clase que se encuentran en los pojos creados anteriormente, enseguida creamos una consulta hql para que me traiga todas las troncales



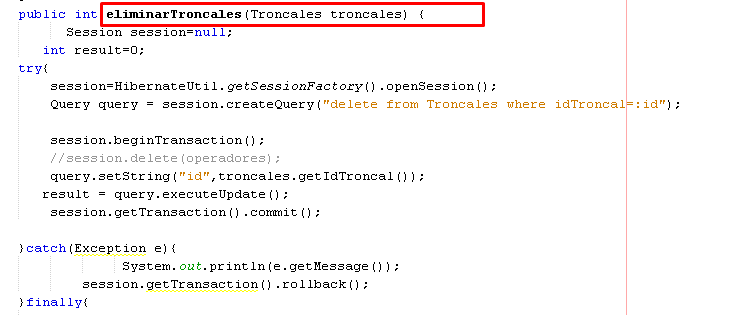
Enseguida configuramos el método de nueva troncal el cual cuenta con un parámetro de troncales para ello creamos una sesión para que me atrape los nuevos registros dentro de la tabla troncales



Ahora creamos el método de modificar troncales la cual le colocamos un parámetro que atrape todas las troncales insertadas enla base de datos, enseguida creamos una sesión la cual va modificar dichos datos utilizando el parámetro update, después de ello guardamos los cambios correspondientes con un commit, en caso que no quiera restablecer a las inserciones anteriores utilizamos el parámetro rollback



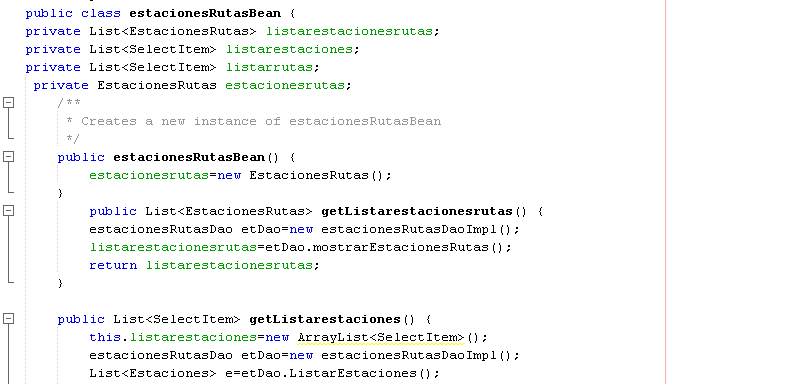
Dentro de esta misma clase generamos el método de eliminar troncales la cual es necesario generar un Query para hacer una consulta la cual tendrá el parámetro de delete y el parámetro por el cual queremos borrar en este caso es por el id de la troncal, y esta al haber sido eliminado debe actualizar los registros dentro de la base de datos



Enseguida creamos el método de buscar en donde en él se crea una consulta hql donde nos mostrara los id de las troncales que están insertadas hasta el momento en la base de datos y como resultado nos arrojara dentro de la consulta los id.



Para poder controlar los métodos creados en las clases anteriores es necesario crear dentro del paquete bean los archivos java server para que controle todos los métodos que se hicieron anterior mente dentro del paquete dao,los cuales son implementados dentro de los objetos es decir los pojos, con la funcionalidad de insertar, modificar, consultar, listar y eliminar



Estop se debe hacer por cada una de as clases creadas tanto para estaciones como troncales y rutas

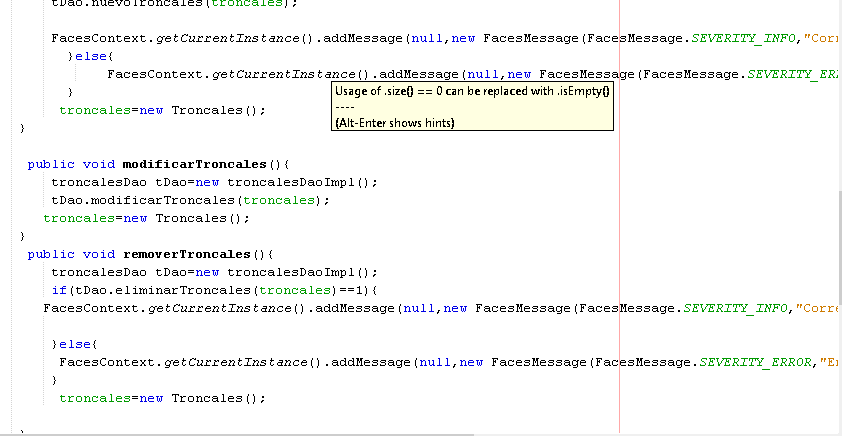


Para la clase ruta



Troncales

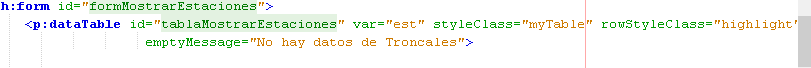
dentro del proceso de remover estación se hace una comparación si existe dicho registro lo elimine si es así que lo elimine de aunó



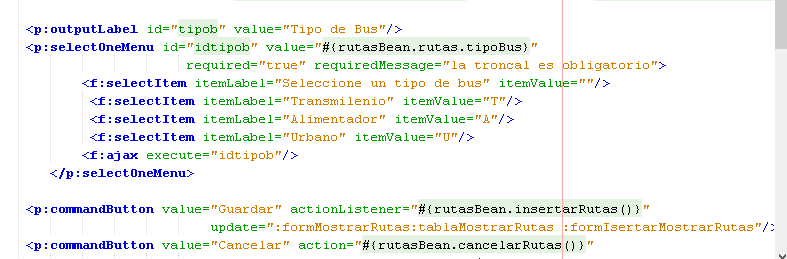
Enseguida vamos a codificar el index de nuestra pagina donde desarrollamos un formulario para cada una de las gestiones de rutas, troncales, estaciones y paradas donde creamos diferentes form para llamar los respectivos métodos creados,



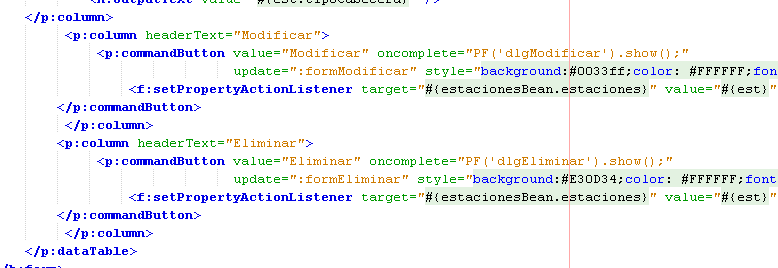
y de igual forma vamos a insertar dentro de estos formularios una tabla para que nos muestre los datos que este contiene insertados dentro de cada tabla de la base de dato.



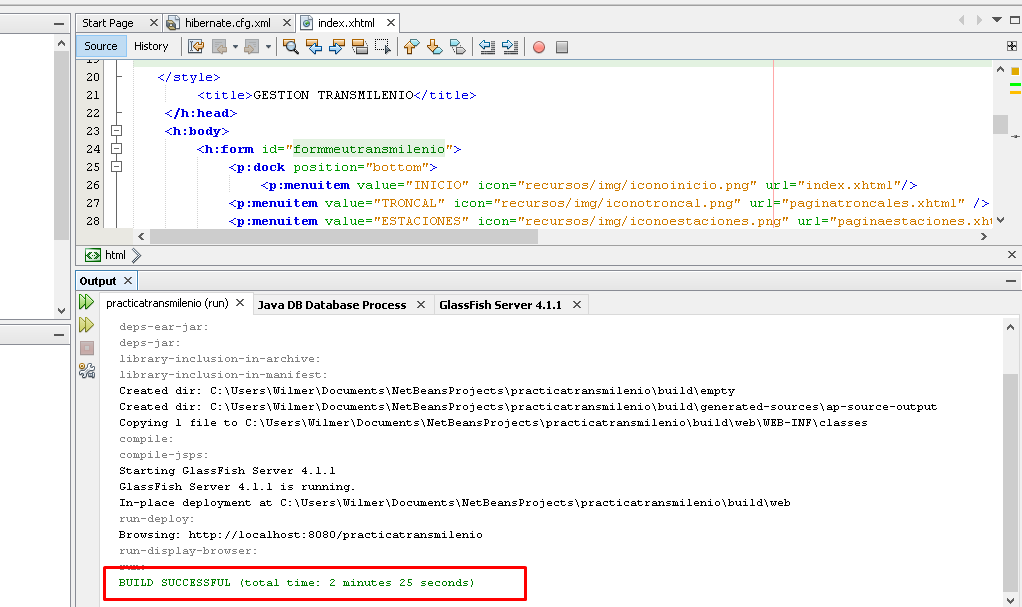
dentro de estos mismos vamos a insertar unas cajas de texto donde se pueda rellenar los datos correspondientes para que se pueda insertar datos dentro de cada una de las gestiones del Transmilenio, este a la ves contendrá dos botones con funcionalidad de guardar y cancelar respectivamente.



Dentro de las gestiones de rutas, troncales, estaciones se encontrara el botón de modificar y eliminar respectivamente dentro de la tabla



Ejecutamos nuestro proyecto



Vamos a nuestro navegador y verificamos que nos esta mostrando la pagina de inicio donde se puede evidenciar la gestión troncal, gestión estaciones gestión rutas, gestión paradas rutas

