



UNIVERSIDAD
DE LOS ANDES
VENEZUELA

INFORME TÉCNICO

MIGRACIÓN Y OPTIMIZACIÓN DEL SISTEMA PROCESO ELECTORAL

Realizado por

Katherine Andrade

Octubre 2018

Mérida, Venezuela

Migración y optimización del sistema Proceso Electoral

Katherine Andrade

Informe Técnico — 23 páginas

Resumen: El presente informe tiene como objetivo dar a conocer de manera concisa y clara las actividades realizadas en Ogangi durante el periodo de las pasantías industriales cortas. Estas tuvieron una duración de ocho (8) semanas.

En este periodo se desarrollaron actividades tales como la creación de las entidades por medio de Hibernate, la creación de la clase que permite la migración de Oracle a Postgresql, la modificación del código del proyecto utilizando todos los conocimientos aprendidos durante la carrera. Para realizar estas actividades se utilizó JAVA, el cual es un lenguaje de programación orientado a Objetos y también se hizo uso de la metodología ágil SCRUM la cual ayudo a tener avances semanales.

Todo esto se realizo junto al tutor industrial el cual siempre estuvo presente para aclarar las dudas que se iban presentando a medida que se avanzaba en el proyecto.

Palabras clave: Pasantías, SCRUM, Programación Orientada a Objetos, Hibernate, migración

Este trabajo fue procesado en L^AT_EX.

Índice general

Introducción	VI
1. Marco de Referencia	1
1.1. Misión:	1
1.2. Visión:	1
2. Bases Teóricas	3
2.1. Programación Orientada a Objetos:	3
2.2. JAVA:	4
2.3. Procedimientos almacenados:	4
2.4. Hibernate:	5
2.5. Sistema de Gestor de Base De Datos(SGBD):	5
3. Aspectos Procedimentales	7
3.1. Plan de trabajo:	7
3.2. Desarrollo del plan:	9
3.2.1. Introducción a la empresa y al entendimiento de los sistemas:	9
3.2.2. Levantamiento de Requerimientos:	9
3.2.3. Análisis:	10
3.2.4. Diseño:	10
3.2.5. Implementación:	10
3.2.6. Pruebas:	11
3.3. Vinculación del proyecto con las materias de la carrera	12
Conclusiones	13

Bibliografías	15
----------------------	-----------

Anexos	16
---------------	-----------

3.4. Creación de procedimientos Almacenados:	17
3.5. Llamados a los procedimientos almacenados:	19
3.6. Entidades en Hibernate:	20
3.7. Archivo XML:	20
3.8. Clase Migrate:	21

Índice de figuras

3.1. Diagrama de Gantt	8
3.2. Corriendo el servidor	11
3.3. Datos consultados por medio de los procedimientos almacenados	11
3.4. SMS recibido con los datos solicitados	12
3.5. Borra a un elector por medio de su cédula	17
3.6. Realiza una consulta diferente dependiendo de una condición	18
3.7. Elimina datos repetidos de una tabla y retorna el resultado	18
3.8. Llamado al procedimiento <code>get_data_info()</code>	19
3.9. El resultado del procedimiento se mostrara dependiendo de la condición que cumple	19
3.10. Entidad que permite crear la tabla cargos	20
3.11. Entidad que permite crear la tabla <code>cc_2018</code>	20
3.12. Se aprecia las conexiones a Oracle y Postgresql	21
3.13. Se aprecian todas las entidades que pueden ser creadas por Hibernate	21
3.14. Se establece la conexión con Oracle y Postgresql	22
3.15. Función que permite la migración de Oracle a Postgresql	22
3.16. Guarda data de Oracle a Postgresql	23
3.17. Finaliza la sesión	23

Introducción

En los últimos años, ha habido un interés creciente en el uso de las tecnologías para la comercialización de productos, haciendo que las empresas estén siempre buscando la manera eficiente de vender los mismos. Para lograr esto, se contratan empresas que se dediquen al desarrollo de tecnologías que permitan cumplir con los objetivos deseados. Ogangi es una empresa que ofrece ese tipo de servicios.

El objetivo de la realización de este proyecto es optimizar código del sistema. Análisis, diseño e implementación de un módulo de Hibernate para el manejador de base de datos.

Ahora bien, para poder llevar a cabo el cumplimiento del objetivo, se necesita de conocimientos obtenidos en el transcurso de la carrera. Estos conocimientos se explican brevemente dentro de este informe

Finalmente, es conveniente acotar la configuración del presente documento, el cual ha sido organizado de la siguiente manera: Capítulo I, marco de referencia, Misión y Visión de la empresa; capítulo II, bases teóricas; capítulo III, aspectos procedimentales, plan de trabajo, desarrollo del plan y vinculación del proyecto con las materias de la carrera; conclusiones, anexos y bibliografías.

Capítulo 1

Marco de Referencia

Ogangi de Venezuela, C.A entrega millones de transacciones a través de las redes de datos móviles de América a través de su plataforma y actualmente procesa más de un millón de mensajes diarios, con sistemas informáticos redundantes, tolerante a fallas y con soporte las 24 horas del día, por 7 días a la semana, los 365 días del año.

1.1. Misión:

Su Misión está representada por la siguiente afirmación: “La tecnología de software y propiedad intelectual de Ogangi acelera y facilita la evolución de servicios móviles de datos. Permite a los operadores celulares crecer más rápidamente su oferta de servicios, a los proveedores y dueños de contenido de crear una presencia móvil, y a los usuarios móviles de gozar de los servicios a través de una experiencia satisfactoria”.

1.2. Visión:

“Los teléfonos, redes y servicios de datos de la telefonía móvil continuarán madurando y evolucionando hasta consolidarse como un medio de comunicación más comparable a los medios tradicionales como televisión y más recientemente Internet. La conveniencia de la ubicuidad y flexibilidad de los servicios móviles de datos permitirá a este nuevo medio inclusive integrar varios de los tradicionales”.

En la actualidad, el negocio de Ogangi se divide principalmente en tres áreas:

- Mensajería corporativa.
- Descarga de contenidos por medio de portales WAP y WEB.
- Desarrollo de aplicaciones móviles.

De las tres áreas mencionadas, la mensajería corporativa ha sido el negocio principal de Ogangi desde sus inicios. Es por eso que año tras año, y juntos con los avances tecnológicos y tendencias en el mercado, Ogangi ha ido enriqueciendo su plataforma de mensajería, ofreciendo a sus clientes cada vez más canales para la entrega de información.

Ogangi cuenta con una solución de mensajería interactiva que le permite a los usuarios obtener información referente al votante en un proceso electoral. Básicamente el cliente (compañía, empresa, etc) encargado del proceso proporciona los datos de los votantes a Ogangi, los cuales son insertados en una base de datos a través de un script, para ser consultados via SMS por los votantes.

Las pasantías fueron realizadas específicamente en el área de desarrollo. Donde se realizó la actualización del proyecto que hace posible el envío de mensajes para obtener información referente al votante en un proceso electoral.

Capítulo 2

Bases Teóricas

2.1. Programación Orientada a Objetos:

La programación orientada a objetos es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.

Los objetos son entidades que combinan estado (atributo), comportamiento (método) e identidad.

El estado de un objeto se refiere al conjunto de atributos y sus valores en un instante de tiempo dado. El comportamiento de un objeto puede modificar el estado de este.

Dicho comportamiento, está directamente relacionado con su funcionalidad y determina las operaciones que este puede realizar o a las que puede responder ante mensajes enviados por otros objetos.

La identidad, es la propiedad que permite diferenciar a un objeto y distinguirse de otros. Generalmente esta propiedad es tal, que da nombre al objeto.

Este tipo de programación, está basada en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

2.2. JAVA:

Java es un lenguaje de programación orientado a objetos y es multiplataforma, ya que se puede ejecutar un mismo código en cualquier SO gracias a que se utiliza un entorno de ejecución de JAVA llamado JRE. Este lenguaje fue desarrollado por Sun Microsystems, posteriormente adquirido por Oracle. En la actualidad puede utilizarse de modo gratuito, pudiéndose conseguir sin problemas un paquete para desarrolladores que oriente la actividad de programar en este lenguaje.

Puede ser modificado por cualquiera, circunstancia que lo convierte en lo que comúnmente se denomina “código abierto”.

2.3. Procedimientos almacenados:

Un procedimiento almacenado es un programa almacenado físicamente en una base de datos. Este programa al ser ejecutado en respuesta a una petición de usuario, es ejecutado directamente en el motor de la base de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

La mayor parte de las veces, se utilizan para encapsular procesos complejos que podrían requerir la ejecución de varias consultas SQL, tales como la manipulación de un conjunto de datos enorme para producir un resultado resumido.

También pueden ser usados para el control de gestión de operaciones, y ejecutar

procedimientos almacenados dentro de una transacción de tal manera que las transacciones sean efectivamente transparentes para ellos.

El hecho de que corra en un servidor separado, aumenta la rapidez de procesamiento de las peticiones del usuario. Los procedimientos almacenados pueden permitir que la lógica del negocio se encuentre como un API en la base de datos, que pueden simplificar la gestión de datos y reducir la necesidad de codificar la lógica en el resto de los programas cliente. Esto puede reducir la probabilidad de que los datos se corrompan por el uso de programas clientes defectuosos o erróneos. De este modo, el motor de base de datos puede asegurar la integridad de los datos y su consistencia con la ayuda de procedimientos almacenados.

2.4. Hibernate:

Hibernate es una herramienta de Mapeo objeto-relacional, esto quiere decir, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

2.5. Sistema de Gestor de Base De Datos(SGBD):

Es un conjunto de programas no visibles que administran y gestionan la información que contiene una base de datos. A través de él se maneja todo acceso a la base de datos con el objetivo de servir de interfaz entre ésta, el usuario y las aplicaciones.

En este proyecto, se utilizaron dos SGBD los cuales fueron: Oracle y Postgresql.

Oracle, es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.

Tradicionalmente, Oracle ha sido el SGBS por excelencia, considerado siempre como el más completo y robusto. Este SGDB siempre ha sido considerado de los más caros, por lo que no se ha estandarizado su uso como otras aplicaciones.

Postgresql, es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, libre y/o apoyada por organizaciones comerciales. La comunidad PostgreSQL se denominada el PGDG (PostgreSQL Global Development Group).

Capítulo 3

Aspectos Procedimentales

3.1. Plan de trabajo:

El proyecto a desarrollar consiste en el levantamiento de información y el análisis del sistema previamente mencionado para posteriormente implementar los mecanismos que permitan finalizar con un producto mas robusto.

Adicionalmente, se debe realizar el levantamiento de información, análisis, diseño e implementación de un modulo de Hibernate para el manejo de base de datos.

- **Objetivos de las pasantías:**

- **Objetivo General:**

- Optimizar código del sistema. Análisis, diseño e implementación de un módulo de Hibernate para el manejador de base de datos.

- **Objetivos Específicos:**

Entre Los objetivos específicos que permitirán que se cumpla con el objetivo general, están:

- Revisión y comprensión de los componentes que conforman la solución actual para el sistema.
- Diseño e implementación de mejoras a los métodos presentes en el sistema.

- Diseño e implementación de un módulo de Hibernate para la gestión de base de datos.
- Actualizar el JDK a la última versión estable.
- Pruebas funcionales del sistema.

En el siguiente Diagrama se puede apreciar el plan de trabajo propuesto por la empresa para las 8 semanas de pasantías y el desarrollo del plan.

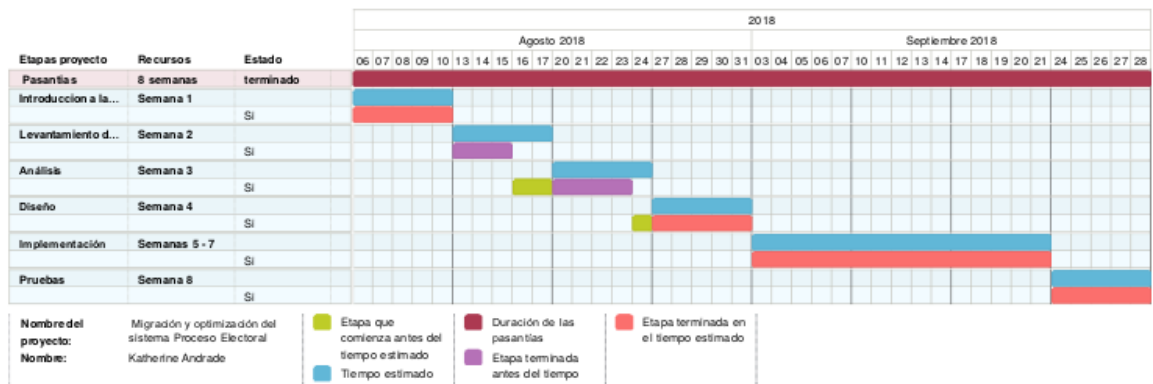


Figura 3.1: Diagrama de Gantt

3.2. Desarrollo del plan:

3.2.1. Introducción a la empresa y al entendimiento de los sistemas:

Semana dedicada a la introducción sobre la implementación actual del sistema, presentada por el Líder de desarrollo José Hidalgo. Explicación del Proyecto que se va a actualizar.

Entrega de una computadora en la cual se realizara dicho proyecto. Instalación de los diferentes proyectos y herramientas a utilizar.

Pruebas simples de las diferentes herramientas necesarias para poder llevar a cabo el proyecto de pasantías, con el fin de entender bien el manejo de dichas herramientas.

3.2.2. Levantamiento de Requerimientos:

Lograr el funcionamiento del proyecto basado en una conexión JPA. Crear las entidades en Hibernate con el fin de facilitar la migración de la base de datos, ya que hibernate permite la migración a cualquier gestor de base de datos.

Creación de las entidades en Hibernate. Se selecciono una clave primaria para cada entidad, ya que a pesar de que la BD original es relacional, esta no contaba con las mismas.

Hibernate no permite crear entidades sin claves primarias. Por lo tanto, se escogió para la mayoría de las tablas una clave primaria autoincremental.

Creación del archivo XML para poder realizar el mapeo.

3.2.3. Análisis:

Modificación de las funciones que se encuentran en la clase CNEUtil, las cuales son las utilizadas para realizar las consultas a la BD que se encuentra en Oracle por medio de una conexión JDBC. Estas fueron modificadas a JPA y su conexión se realiza por medio de Hibernate.

Creación de Procedimientos Almacenados en Oracle de las consultas que se realizaban por medio de JDBC.

Comienzo de la clase con la cual se realizara la migración de Oracle a Postgresql.

3.2.4. Diseño:

Creación de las tablas en Postgresql por medio del mapeo objeto-relacional. Esta creación se hizo en el servidor local. También se realizó la creación de los mismos Procedimientos Almacenados que se encuentran en Oracle, ya que estos funcionan correctamente.

Culminación de la clase Migrate, la cual es la encargada de realizar la migración de Oracle a Postgresql.

3.2.5. Implementación:

Modificación de las funciones de las clases Updater y getDBDataTest. Cambio de JDBC a JPA. Las consultas se realizaron por medio de Procedimientos almacenados.

La clase Updater consta de funciones encargadas de insertar valores en una tabla específica, actualizar o eliminar valores repetidos de la misma. La clase getDBDataTest hace una búsqueda en la BD de una cédula con el fin de probar que la conexión es exitosa.

3.2.6. Pruebas:

Realización de la migración de Oracle a Postgresql.

Envío de un mensaje de texto para probar el funcionamiento del proyecto conectado con la BD remota que se encuentra en Oracle luego de terminar todas las modificaciones del proyecto.

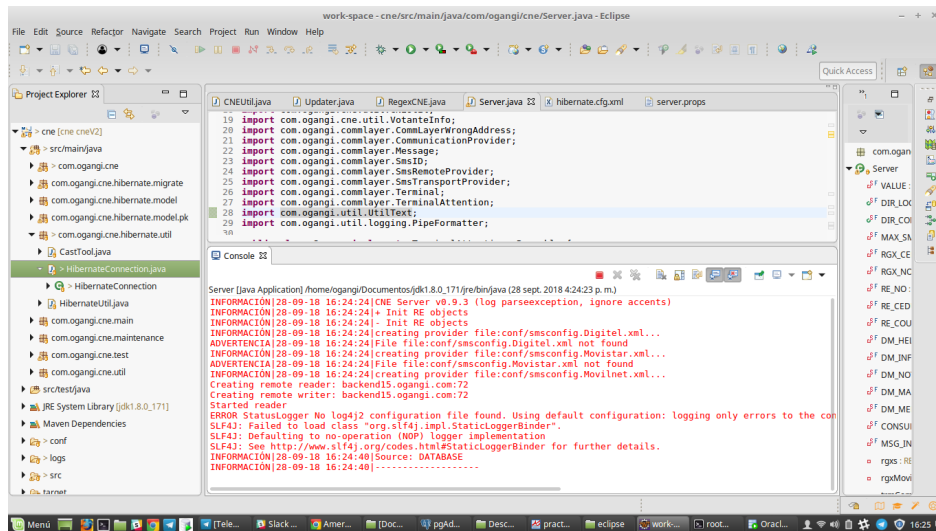


Figura 3.2: Corriendo el servidor

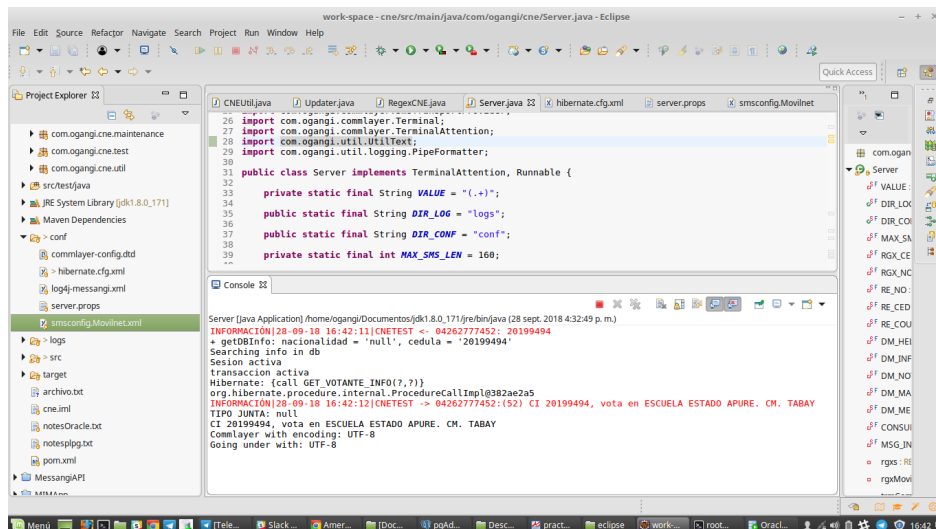


Figura 3.3: Datos consultados por medio de los procedimientos almacenados

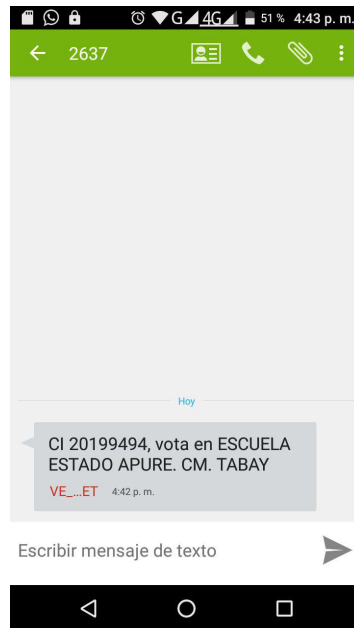


Figura 3.4: SMS recibido con los datos solicitados

3.3. Vinculación del proyecto con las materias de la carrera

Para poder llevar a cabo el proyecto, fue necesario el uso de conocimientos teóricos y prácticos que han sido impartidos en el transcurso de la carrera. Uno de los mas importantes es el conocimiento adquirido de programación, donde se desarrolla la lógica para resolver este tipo de problemas.

En el transcurso de la carrera se ven 3 programaciones, cada una con su nivel de dificultad. Para desarrollar este proyecto se necesito específicamente de los conocimientos aprendidos en Programación 2 y 3.

En programación 2, se aprendió sobre el manejo de archivos y la programación orientada a Objetos. Este ultimo concepto es muy importante, ya que el lenguaje de programación utilizado para el desarrollo fue JAVA, el cual es un lenguaje de programación orientado a objetos.

El manejo de archivos fue necesario para guardar los datos extraídos de la base de datos en dichos documentos.

En programación 3, se aprendió sobre estructuras de datos, ya que estas permiten organizar los datos de una manera eficiente, logrando así hacer un programa óptimo.

Otros conceptos utilizados fueron los adquiridos en las materias del profesional, estas son: Base de Datos e Ingeniería del Software.

En Base de Datos, se aprendió como diseñar de manera eficiente bases de datos relacionales. También como realizar la operaciones: insertar, modificar, eliminar y consultar a una Base de Datos.

Otros de los conocimientos adquiridos, fue el de como manejar un Sistema de Gestor de Base De Datos, en este caso Postgresql, el cual es uno de los sistemas que esta siendo utilizado por la empresa.

La creación y uso de los Procedimientos almacenados también se aprende en esta materia. Dichos procedimientos fueron muy utilizados en el proyecto.

En ingeniería del Software, se enseñan los principios y las metodologías para el desarrollo y mantenimiento de sistemas software (Zelkovitz, 1978). En este proyecto se utilizo Scrum, la cual es una metodología ágil que se basa en una estructura de desarrollo incremental.

Conclusiones

Comúnmente los estudiantes egresados de las universidades no han tenido la oportunidad de conocer el campo laboral para el que han sido formados, por esta razón realizar pasantías resulta una buena oportunidad para adquirir experiencia laboral y poder aplicar todo lo aprendido durante la carrera, guiado por un tutor empresarial el cual siempre ayudará al pasante en todo el periodo de duración de las mismas.

El conocimiento adquirido te permite crecer profesionalmente y sentir mas seguridad a la hora de enfrentar el campo laboral.

El principal objetivo del presente proyecto era optimizar el código del sistema y realizar el análisis, diseño e implementación de un módulo de Hibernate para el manejador de base de datos el cuál se logro completar satisfactoriamente, aplicando todos los conocimientos adquiridos en la carrera.

Una acotación importante sería mejorar la base de datos utilizada por este proyecto ya que no cuenta con unas relaciones que permitan hacer las búsquedas de una manera eficiente.

Realizar pasantías en Ogangi, es una experiencia gratificante ya que aparte de todos los conocimientos adquiridos, el ambiente de trabajo es muy agradable. Hay mucha unión y compañerismo y en cualquier situación en la que se necesite ayuda para realizar algún objetivo de un proyecto se puede contar con el apoyo de todos.

Ogangi resulta ser una empresa en la cual la unión entre sus empleados hace que el trabajo sea mas agradable.

Bibliografía

- [1] Wikilibros(2014) *Programación Orientada a Objetos*,
https://es.wikibooks.org/wiki/Programación_Orientada_a_Objeto
- [2] Revista Digital *INESEM Los gestores de bases de datos más usados*,
<https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/>
- [3] Wikipedia(2018), *Objeto(programación)*,
[https://es.wikipedia.org/wiki/Objeto_\(programación\)#Estado](https://es.wikipedia.org/wiki/Objeto_(programación)#Estado)
- [4] educación IT blog(2013) *¿Qué es Java Hibernate?*,
<http://blog.educacionit.com/2013/02/07/que-es-java-hibernate/>
- [5] Wikipedia(2017), *Hibernate*,
<https://es.wikipedia.org/wiki/Hibernate>
- [6] Wikipedia(2018), *Procedimiento almacenado*,
https://es.wikipedia.org/wiki/Procedimiento_almacenado
- [7] Definicion, *Definición de Java*,
<https://definicion.mx/java/>

Anexos

Las siguientes imágenes muestra código realizado por mi persona.

3.4. Creación de procedimientos Almacenados:

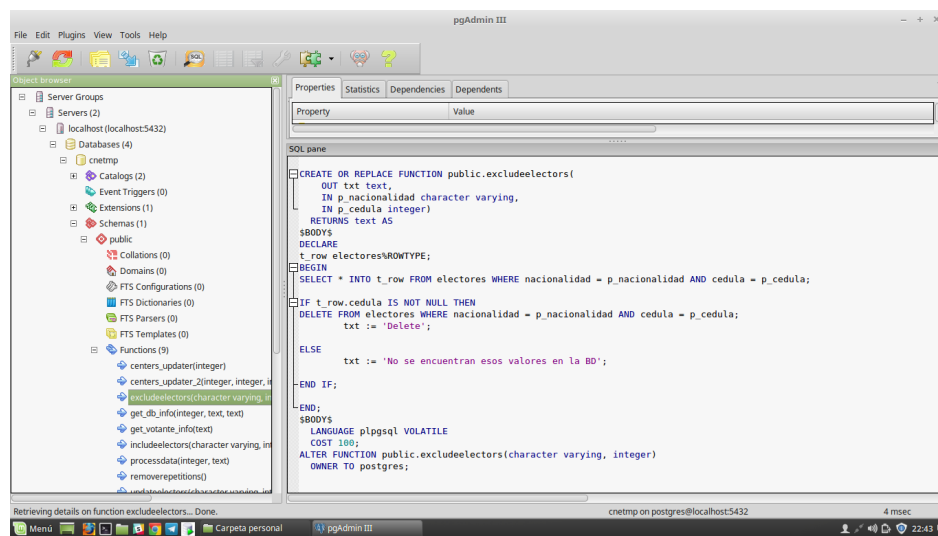


Figura 3.5: Borra a un elector por medio de su cédula

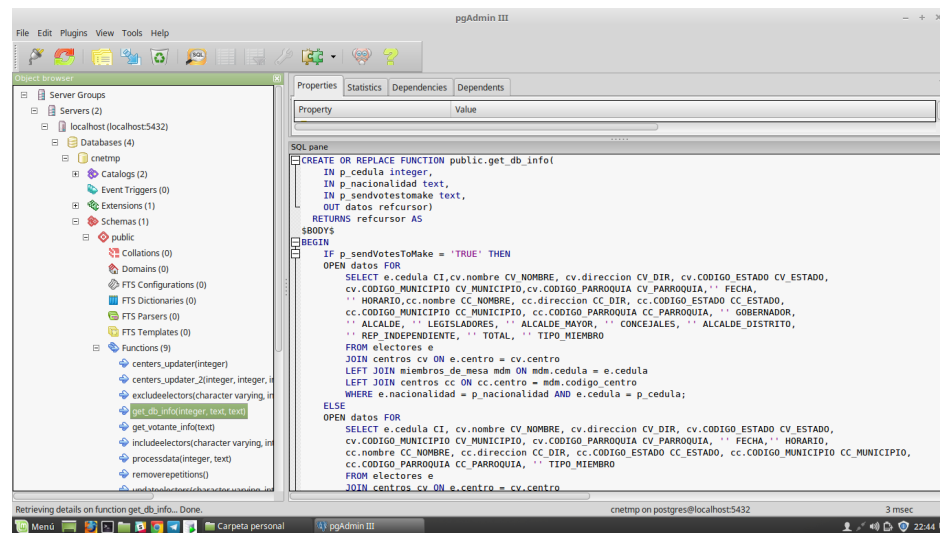


Figura 3.6: Realiza una consulta diferente dependiendo de una condición

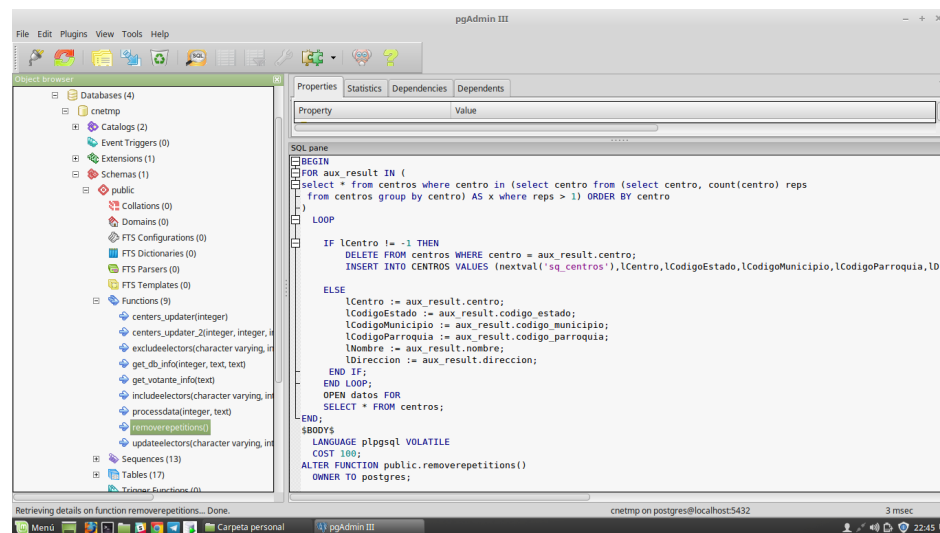
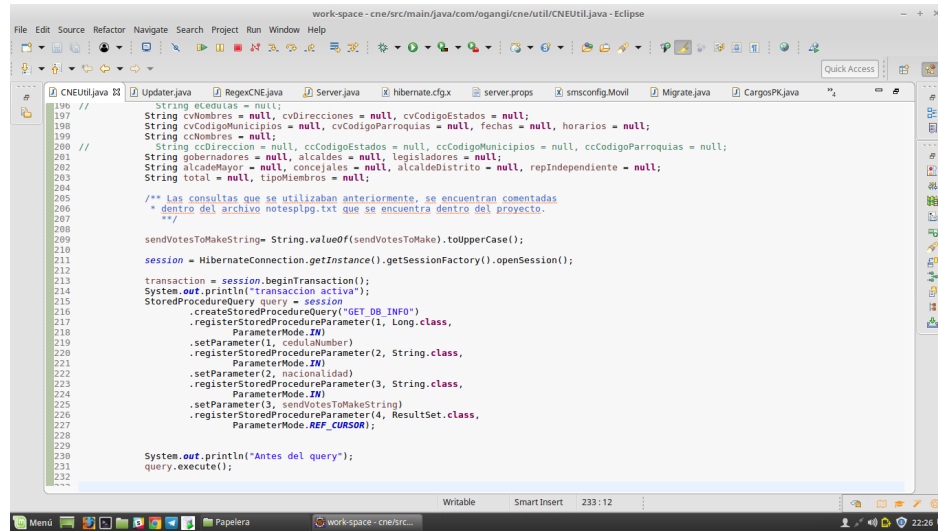


Figura 3.7: Elimina datos repetidos de una tabla y retorna el resultado

3.5. Llamados a los procedimientos almacenados:

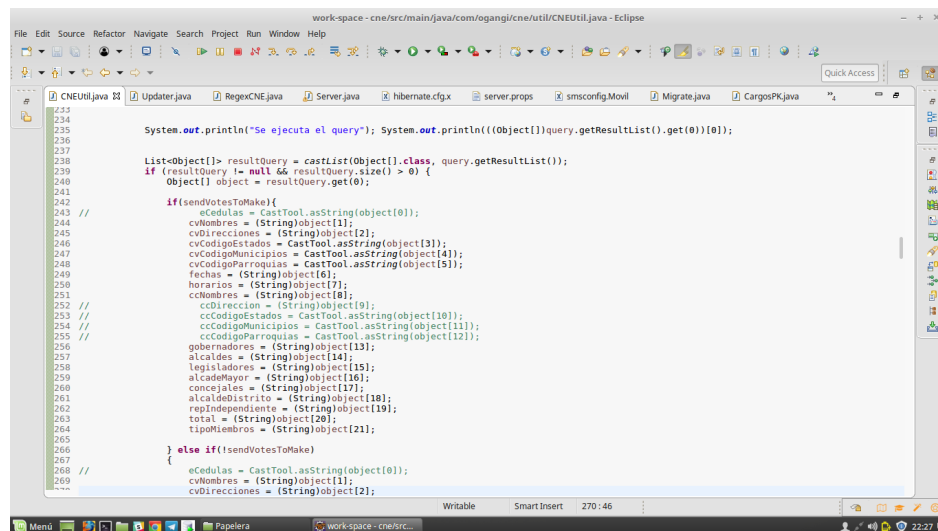


```

190 //
191 String eCedulas = null;
192 String cvNombres = null, cvDirecciones = null, cvCodigoEstados = null;
193 String cvCodigoMunicipios = null, cvCodigoParroquias = null, fechas = null, horarios = null;
194 String ccNombres = null;
195 //
196 String ccDireccion = null, ccCodigoEstados = null, ccCodigoMunicipios = null, ccCodigoParroquias = null;
197 String gobernadores = null, alcaldes = null, legisladores = null;
198 String alcaldeMayor = null, concejales = null, alcaldeDistrito = null, repIndependiente = null;
199 String total = null, tipoMiembros = null;
200 //
201 /** Las consultas que se utilizaban anteriormente, se encuentran comentadas
202  * dentro del archivo notesplg.txt que se encuentra dentro del proyecto.
203  */
204
205 sendVotesToMakeString = String.valueOf(sendVotesToMake).toUpperCase();
206
207 session = HibernateConnection.getInstance().getSessionFactory().openSession();
208
209 transaction = session.beginTransaction();
210 System.out.println("transaccion activa");
211 StoredProcedureQuery query = session
212     .createStoredProcedureQuery("GET_DB_INFO")
213     .registerStoredProcedureParameter(1, Long.class,
214         ParameterMode.IN)
215     .setParameter(1, cedulaNumber)
216     .registerStoredProcedureParameter(2, String.class,
217         ParameterMode.IN)
218     .setParameter(2, nacionalidad)
219     .registerStoredProcedureParameter(3, String.class,
220         ParameterMode.IN)
221     .setParameter(3, sendVotesToMakeString)
222     .registerStoredProcedureParameter(4, ResultSet.class,
223         ParameterMode.REF_CURSOR);
224
225 System.out.println("Antes del query");
226 query.execute();
227
228
229
230
231
232

```

Figura 3.8: Llamado al procedimiento get_data_info()



```

233
234 System.out.println("Se ejecuta el query"); System.out.println(((Object[])query.getResultList().get(0))[0]);
235
236
237
238 List<Object[]> resultQuery = castList(Object[] class, query.getResultList());
239 if (resultQuery != null && resultQuery.size() > 0) {
240     Object[] object = resultQuery.get(0);
241
242     if (sendVotesToMake) {
243         eCedulas = CastTool.asString(object[0]);
244         cvNombres = (String)object[1];
245         cvDirecciones = (String)object[2];
246         cvCodigoEstados = CastTool.asString(object[3]);
247         cvCodigoMunicipios = CastTool.asString(object[4]);
248         cvCodigoParroquias = CastTool.asString(object[5]);
249         fechas = (String)object[6];
250         horarios = (String)object[7];
251         ccNombres = (String)object[8];
252         ccDireccion = (String)object[9];
253         ccCodigoEstados = CastTool.asString(object[10]);
254         ccCodigoMunicipios = CastTool.asString(object[11]);
255         ccCodigoParroquias = CastTool.asString(object[12]);
256         gobernadores = (String)object[13];
257         alcaldes = (String)object[14];
258         legisladores = (String)object[15];
259         alcaldeMayor = (String)object[16];
260         concejales = (String)object[17];
261         alcaldeDistrito = (String)object[18];
262         repIndependiente = (String)object[19];
263         total = (String)object[20];
264         tipoMiembros = (String)object[21];
265     } else if (!sendVotesToMake) {
266         eCedulas = CastTool.asString(object[0]);
267         cvNombres = (String)object[1];
268         cvDirecciones = (String)object[2];
269
270
271
272

```

Figura 3.9: El resultado del procedimiento se mostrara dependiendo de la condición que cumple

3.6. Entidades en Hibernate:

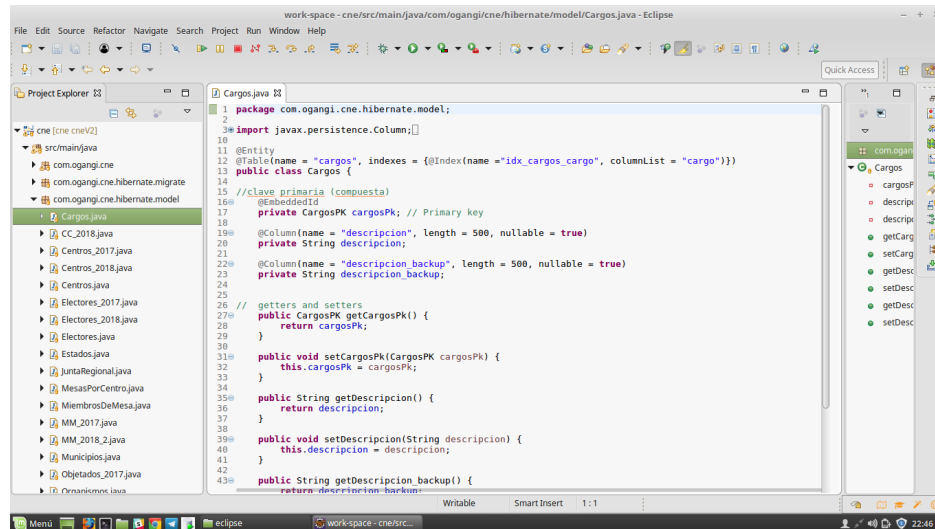


Figura 3.10: Entidad que permite crear la tabla cargos

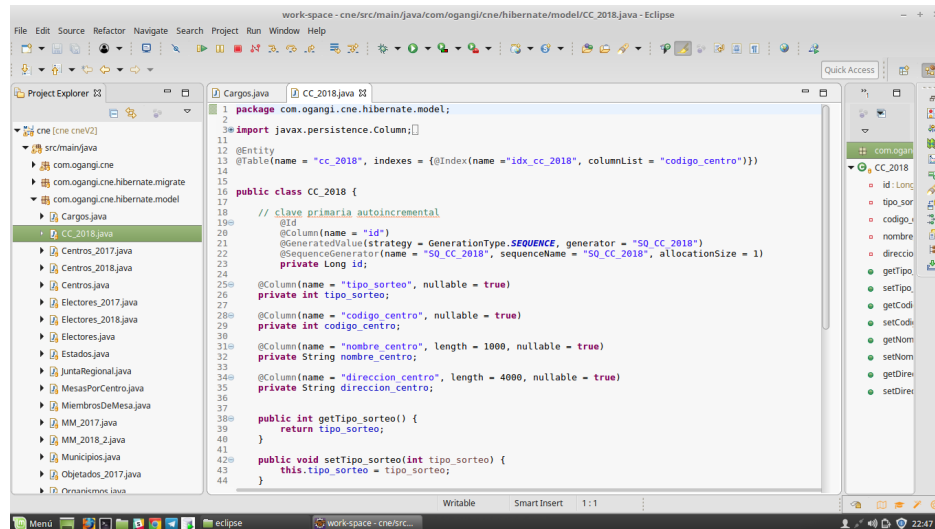
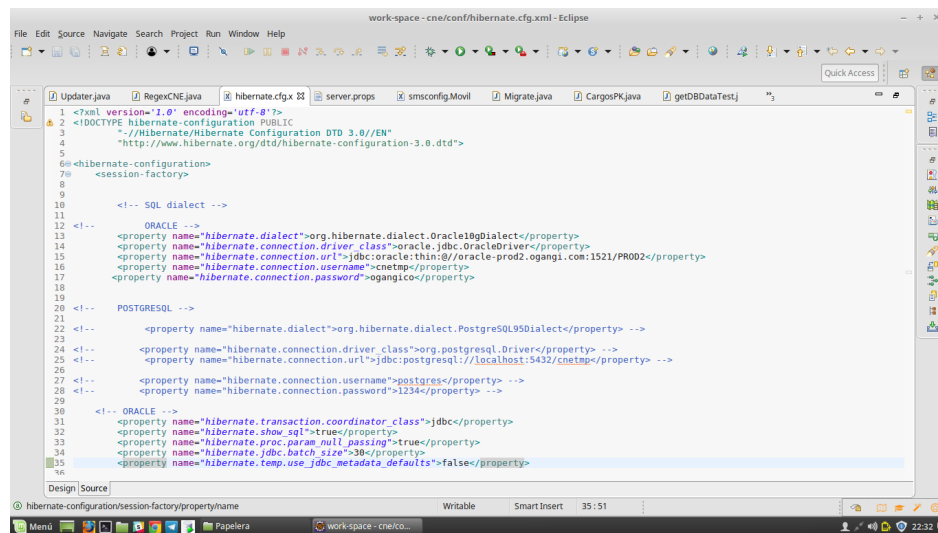


Figura 3.11: Entidad que permite crear la tabla cc_2018

3.7. Archivo XML:

Este archivo se creó junto al tutor empresarial

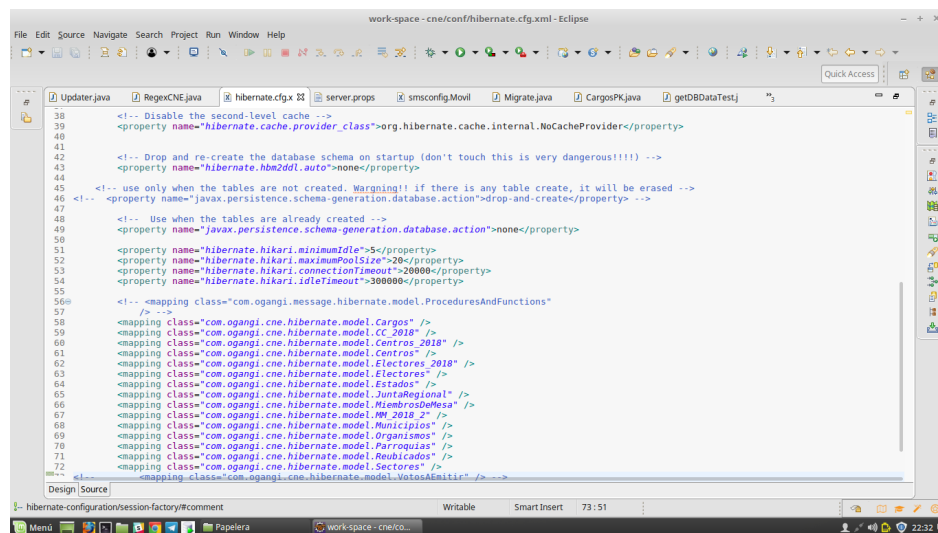


```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3     "-//Hibernate/Hibernate Configuration DTD 3.0/EN"
4     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5
6 <hibernate-configuration>
7     <session-factory>
8
9
10     <!-- SQL dialect -->
11
12     <!--
13     <property name="hibernate.dialect">org.hibernate.dialect.Oracle10gDialect</property>
14     <property name="hibernate.connection.driver_class">oracle.jdbc.OracleDriver</property>
15     <property name="hibernate.connection.url">jdbc:oracle:thin:@oracle-prod2.ogangi.com:1521/PROD2</property>
16     <property name="hibernate.connection.username">cnctmp</property>
17     <property name="hibernate.connection.password">ogangico</property>
18
19     <!--
20     <!-- PostgreSQL -->
21
22     <!--
23     <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQL95Dialect</property> -->
24     <!--
25     <property name="hibernate.connection.driver_class">org.postgresql.Driver</property> -->
26     <!--
27     <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/cnctmp</property> -->
28     <!--
29     <property name="hibernate.connection.username">postgres</property> -->
30     <!--
31     <property name="hibernate.connection.password">1234</property> -->
32
33     <!-- ORACLE -->
34     <property name="hibernate.transaction.coordinator_class">jdbc</property>
35     <property name="hibernate.show_sql">>true</property>
36     <property name="hibernate.proc_param_null_passing">>true</property>
37     <property name="hibernate.jdbc.batch_size">38</property>
38     <property name="hibernate.temp_use_jdbc_metadata_defaults">false</property>
39
40     </session-factory>
41 </hibernate-configuration>

```

Figura 3.12: Se aprecia las conexiones a Oracle y Postgresql



```

38 <!-- Disable the second-level cache -->
39 <property name="hibernate.cache.provider_class">org.hibernate.cache.internal.NoCacheProvider</property>
40
41
42 <!-- Drop and re-create the database schema on startup (don't touch this is very dangerous!!!!) -->
43 <property name="hibernate.hbm2ddl.auto">none</property>
44
45 <!-- Use only when the tables are not created. Warning!! if there is any table create, it will be erased -->
46 <!-- <property name="javax.persistence.schema-generation.database.action">drop-and-create</property> -->
47
48 <!-- Use when the tables are already created -->
49 <property name="javax.persistence.schema-generation.database.action">none</property>
50
51 <property name="hibernate.hikari.minimumIdle">5</property>
52 <property name="hibernate.hikari.maximumPoolSize">20</property>
53 <property name="hibernate.hikari.connectionTimeout">20000</property>
54 <property name="hibernate.hikari.idleTimeout">300000</property>
55
56 <!--
57 <!--
58 <mapping class="com.ogangi.cne.hibernate.model.Cargos" />
59 <mapping class="com.ogangi.cne.hibernate.model.CC_2018" />
60 <mapping class="com.ogangi.cne.hibernate.model.Centros" />
61 <mapping class="com.ogangi.cne.hibernate.model.Electores" />
62 <mapping class="com.ogangi.cne.hibernate.model.Electores_2018" />
63 <mapping class="com.ogangi.cne.hibernate.model.Electores" />
64 <mapping class="com.ogangi.cne.hibernate.model.Estados" />
65 <mapping class="com.ogangi.cne.hibernate.model.JuntaRegional" />
66 <mapping class="com.ogangi.cne.hibernate.model.MiembrosMesa" />
67 <mapping class="com.ogangi.cne.hibernate.model.PW_2018_2" />
68 <mapping class="com.ogangi.cne.hibernate.model.Municipios" />
69 <mapping class="com.ogangi.cne.hibernate.model.Organismos" />
70 <mapping class="com.ogangi.cne.hibernate.model.Parrquias" />
71 <mapping class="com.ogangi.cne.hibernate.model.Reubicados" />
72 <mapping class="com.ogangi.cne.hibernate.model.Sectores" />
73 <!--
74 <!--
75 <mapping class="com.ogangi.cne.hibernate.model.VotosAEntier" /> -->
76 </mapping>

```

Figura 3.13: Se aprecian todas las entidades que pueden ser creadas por Hibernate

3.8. Clase Migrate:

Se mostrará algunos segmentos de una de las funciones del código de la clase migrate, la cual fue realizada por mi persona.

```

import com.ogangi.cne.hibernate.util.HibernateConnection;
import com.ogangi.util.UtilBD;

public class Migrate {

    static Session session = null;
    private static DataSource dataSource;

    public String getProperty(Properties config, String propName) throws Exception {
        String value = null;
        value = config.getProperty(propName);
        if (value == null)
            throw new Exception("property " + propName + " not found in file");
        return value;
    }

    public Migrate() throws Exception {
        Properties config = new Properties();
        String fullName = System.getProperty("user.dir") + System.getProperty("file.separator") +
"conf"
            + System.getProperty("file.separator") + "server.props";
        FileInputStream file = new FileInputStream(fullName);
        config.load(file);
        int nConnections;
        try {
            nConnections = Integer.parseInt(getProperty(config, "connections"));
        } catch (Exception e) {
            nConnections = 20;
        }

        dataSource = UtilBD.getDBCPooledSource( getProperty(config, "DBurl"),getProperty(config,
"userName"), getProperty(config, "password"), nConnections, -1);
    }
}

```

Figura 3.14: Se establece la conexión con Oracle y Postgresql

```

// TODO Búsqueda de la tabla Centros
public void getMigrateCentros() throws Exception {

    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    long timerTime = 0;
    Transaction transaction = null;

    try {
        session = HibernateConnection.getInstance().getSessionFactory().openSession();
        System.out.println("session activa");
        transaction = session.beginTransaction();

        Centros centro = new Centros();
        Centros centroSql = new Centros();

        conn = dataSource.getConnection();
        String query = "";

        query = "SELECT * FROM centros";

        ps = conn.prepareStatement(query);

        rs = ps.executeQuery();

        int count = 0;
        System.out.println("antes de buscar en oracle");
        System.out.println("tamaño " + rs.getMetaData().getColumnCount());
    }
}

```

Figura 3.15: Función que permite la migración de Oracle a Postgresql

