Javier Díaz Frías –Javier Oblaré Liébana –Alejandro Escobar Martin

.

TG3

HIBERNATE VS ENTITY FRAMEWORK: COMPARACION ORM

Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc450671190)

[1.1 Autores 3](#_Toc450671191)

[1.2 Planificación 3](#_Toc450671192)

[1.3 Entrega 3](#_Toc450671193)

[2. Requisitos del prototipo a implementar 4](#_Toc450671194)

[2.1 Requisitos funcionales 4](#_Toc450671195)

[2.2 Otros requisitos 4](#_Toc450671196)

[3. Criterios de comparación en la implementación 5](#_Toc450671197)

[3.1 Criterio 1: Dificultad de aprendizaje 5](#_Toc450671198)

[3.2 Criterio 2: Número de líneas de código realizadas 5](#_Toc450671199)

[3.3 Criterio 3: Velocidad de inserción y consulta de datos 5](#_Toc450671200)

[3.4 Criterio 4: Base de datos 5](#_Toc450671201)

[3.5 Criterio 5: Peso Aplicación 6](#_Toc450671202)

[4. Proyecto de implementación de un prototipo del sistema utilizando Hibernate 7](#_Toc450671203)

[4.1 Documentación de diseño 7](#_Toc450671204)

[4.2 Documentación de construcción 8](#_Toc450671205)

[4.3 Documentación de pruebas 15](#_Toc450671206)

[4.4 Documentación de instalación 16](#_Toc450671207)

[4.5 Manual de usuario 16](#_Toc450671208)

[5. Proyecto de implementación de un prototipo del sistema utilizando Entity Framework](#_Toc450671209)

[5.1 Documentación de diseño 17](#_Toc450671210)

[5.2 Documentación de construcción 19](#_Toc450671211)

[5.3 Documentación de pruebas 23](#_Toc450671214)

[6. Comparación de las dos implementaciones 24](#_Toc450671215)

[6.1 Evaluación de los criterios en la implementación usando la tecnología Hibernate 24](#_Toc450671216)

[6.2 Evaluación de los criterios en la implementación usando la tecnología Entity Framework 25](#_Toc450671217)

[7. Comparación de la implementación de las tecnologías 25](#_Toc450671218)

[8. Conclusiones 27](#_Toc450671219)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Somos el grupo 3 del turno de tarde, integrado por Javier Oblaré Liébana (coordinador), Javier Díaz Frías y Alejandro Escobar Martin

## 1.2 Planificación

Hemos utilizado una herramienta online de diagramas (GanttPro, versión gratuita), para la planificación del trabajo, repartiéndose las tareas del trabajo, con su número de horas de trabajo por cada una.

Este enlace le llevará a dicha planificación:

<https://app.ganttpro.com/shared/token/1e5030e2bdfe0efe1840353103f33bcd4f238cc069ab5e9b47a48359b729f2d6>

## 1.3 Entrega

Hemos ido realizando el trabajo en el repositorio GitHub, dentro del cual su URL es:

<https://github.com/JaviOblare/TG3.git>

# 2. Requisitos del prototipo a implementar

## 2.1 Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RF01 | Página de Inicio – Página inicial de la aplicación con dos botones. |
| RF02 | Ingresar nuevos empleados – La aplicación debe dejar introducir nuevos empleados a la base de datos (botón nuevo empleado) |
| RF03 | La conexión a la base de datos deber ser mediante el mapeo de la tecnología empleada (Hibernate/Entity Framework) |
| RF04 | Página empleados – Listado con información de los empleados de la empresa (botón empleados desde la página inicial). |
| RF05 | Mensaje – Una ver introducido un mensaje debe saltar un mensaje confirmando su introducción. |

## 2.2 Otros requisitos

| **REQ.** | **DESCRIPCIÓN** |
| --- | --- |
| RNF01 | Interfaz de Usuario -- Interfaz de usuario sencilla para la fácil utilización de cualquiera. |
| RNF02 | Tiempo de respuesta – Que la aplicación tenga un buen tiempo de respuesta |
|  |  |

# 3. Criterios de comparación en la implementación

## 3.1 Criterio 1: Dificultad de aprendizaje

Nombre del criterio: Dificultad de aprendizaje.

Descripción: Dificultad de aprendizaje de uso de la tecnología.

Tipo de valor: Numérico (0-10).

## 3.2 Criterio 2: Número de líneas de código realizadas

Nombre del criterio: Número de líneas.

Descripción: Número de líneas de código realizadas.

Tipo de valor: Numérico.

## 3.3 Criterio 3: Velocidad de inserción y consulta de datos

Nombre del criterio: Velocidad de consulta de datos.

Descripción Velocidad de consulta de los datos de empleados registrados en la base de datos.

Tipo de valor: Numérico.

## 3.4 Criterio 4: Base de datos

Nombre del criterio: Base de datos.

Descripción: Con que sistema se realiza la base de datos de la aplicación.

Tipo de valor: texto.

## 3.5 Criterio 5: Peso Aplicación

Nombre del criterio: Peso Aplicación.

Descripción: Peso en bytes de la aplicación desarrollada.

Tipo de valor: numérico

**3.6 Criterio 6: Look and Feel**

Nombre del criterio: Look and Feel.

Descripción: Look and Feel de las interfaces de la aplicación.

Tipo de valor: Texto

**3.7 Criterio 7: Uso de CPU**

Nombre del criterio: Uso de CPU

Descripción: Uso máximo de CPU de la aplicación

Tipo de valor: Porcentaje

**3.8 Criterio 8: Uso de memoria**

Nombre del criterio: Uso de memoria

Descripción: Uso de memoria por parte de la aplicación

Tipo de valor: Numérico

**3.9 Criterio 9: Complejidad ciclomatica**

Nombre del criterio: Complejidad ciclomatica

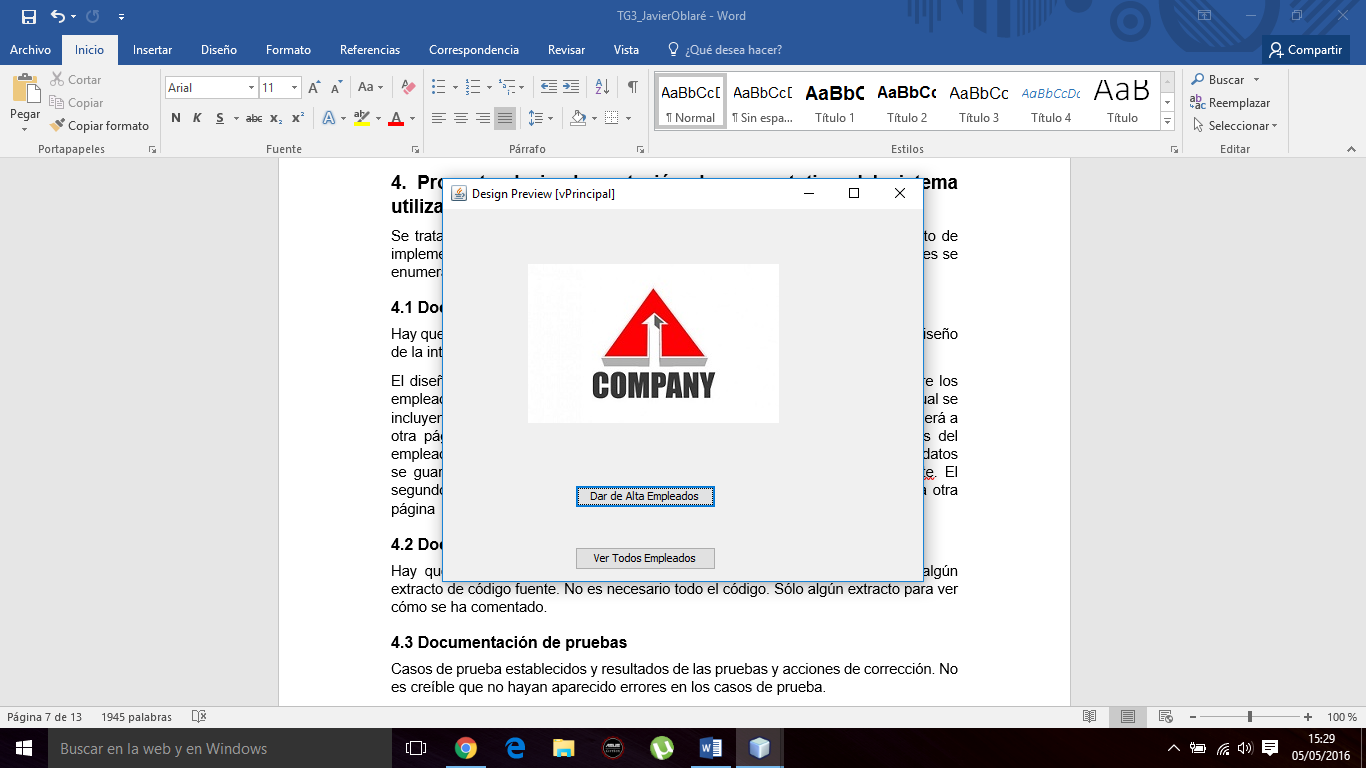
Descripción: Medición cuantitativa de la complejidad lógica de la aplicación

Tipo de valor: Numérico

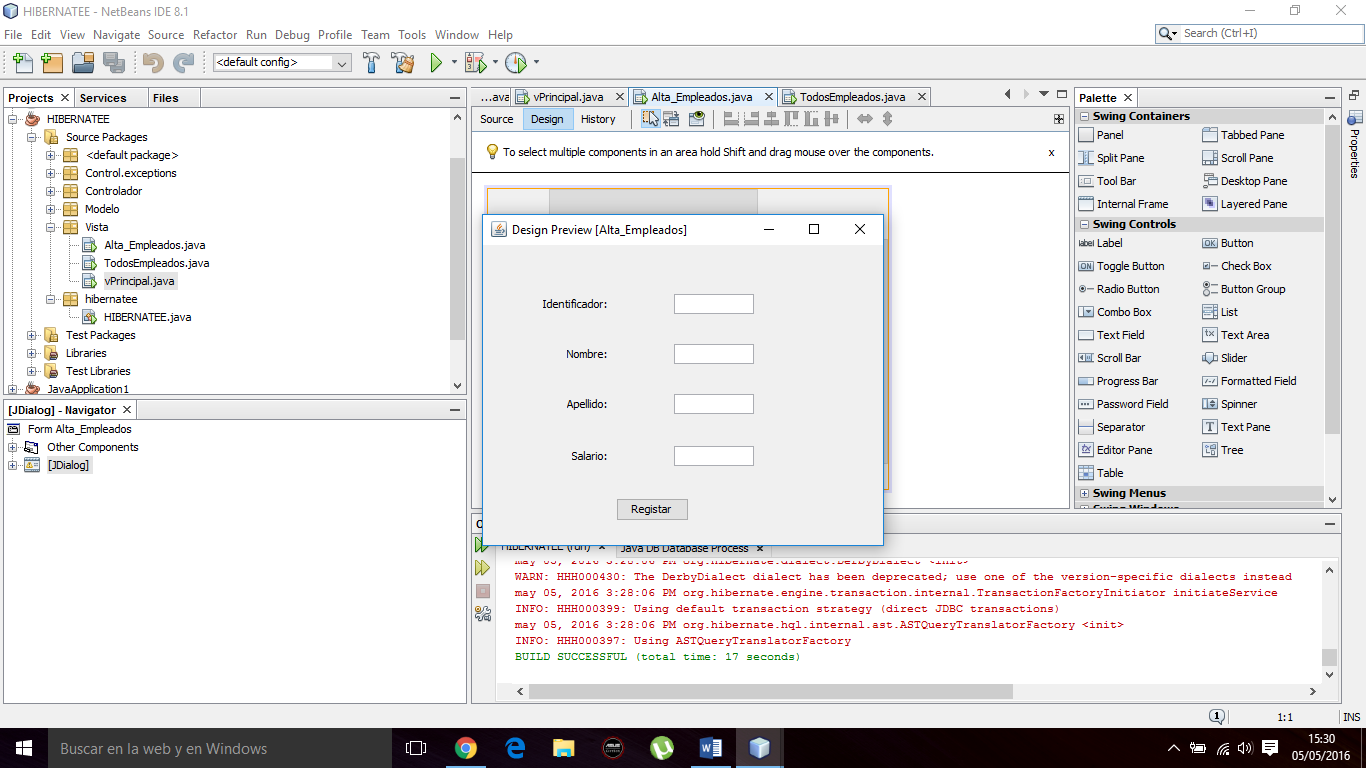
# 4. Proyecto de implementación de un prototipo del sistema utilizando Hibernate

## 4.1 Documentación de diseño

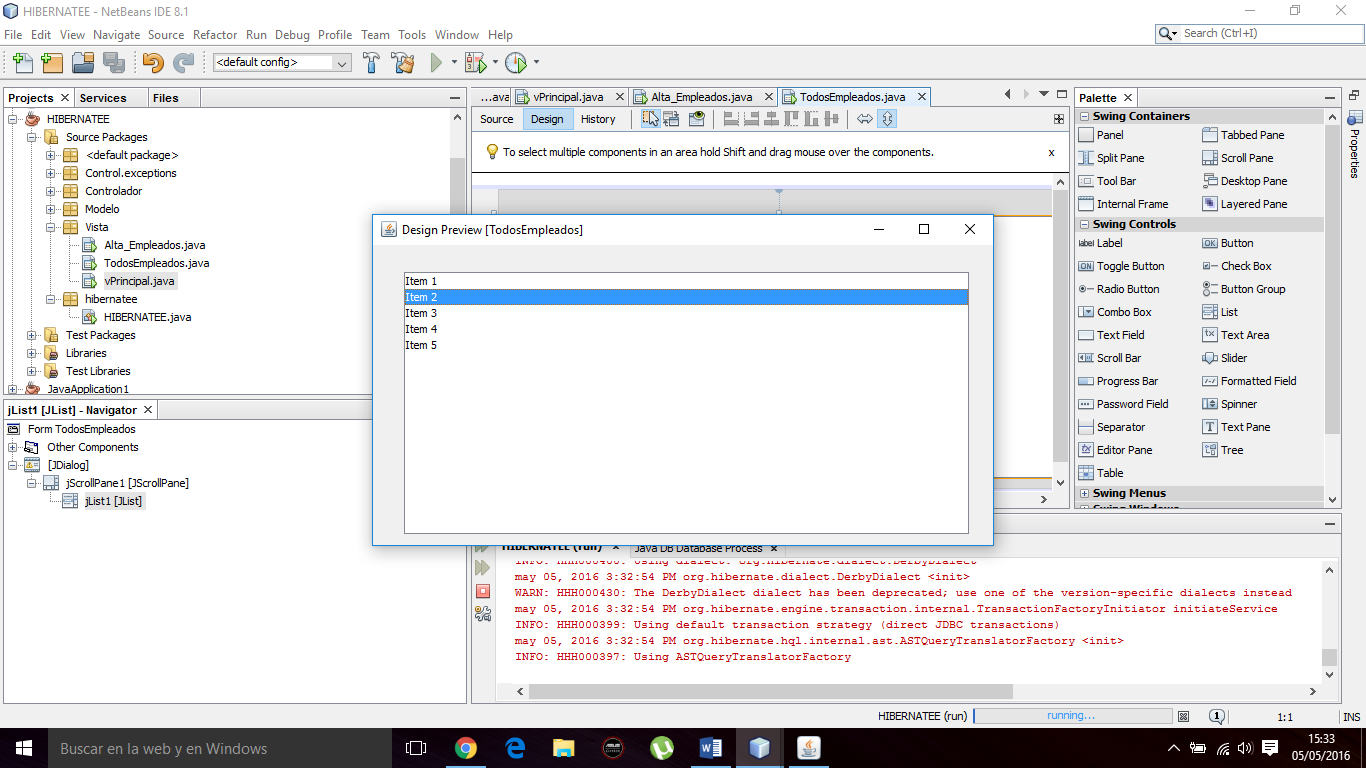
El diseño de la aplicación basada en Hibernate consiste en una aplicación sobre los empleados de una empresa. Dicha aplicación consta de una página inicial en la cual se incluyen dos botones. Un primer botón llamado “Dar de Alta Empleados” el cuál accederá a otra página, dentro de la cuál tendremos un formulario que rellenar los datos del empleado, el registro se confirmará con un botón llamado “Registrar”, con lo que los datos se guardaran en la base de datos creada anteriormente a través de Hibernate. El segundo botón de la página inicial llamado “Ver Todos Empleados”, el cuál accederá a otra página que nos mostrará todos los empleados anteriormente registrados.



Página inicial



Página Dar Alta Empleados

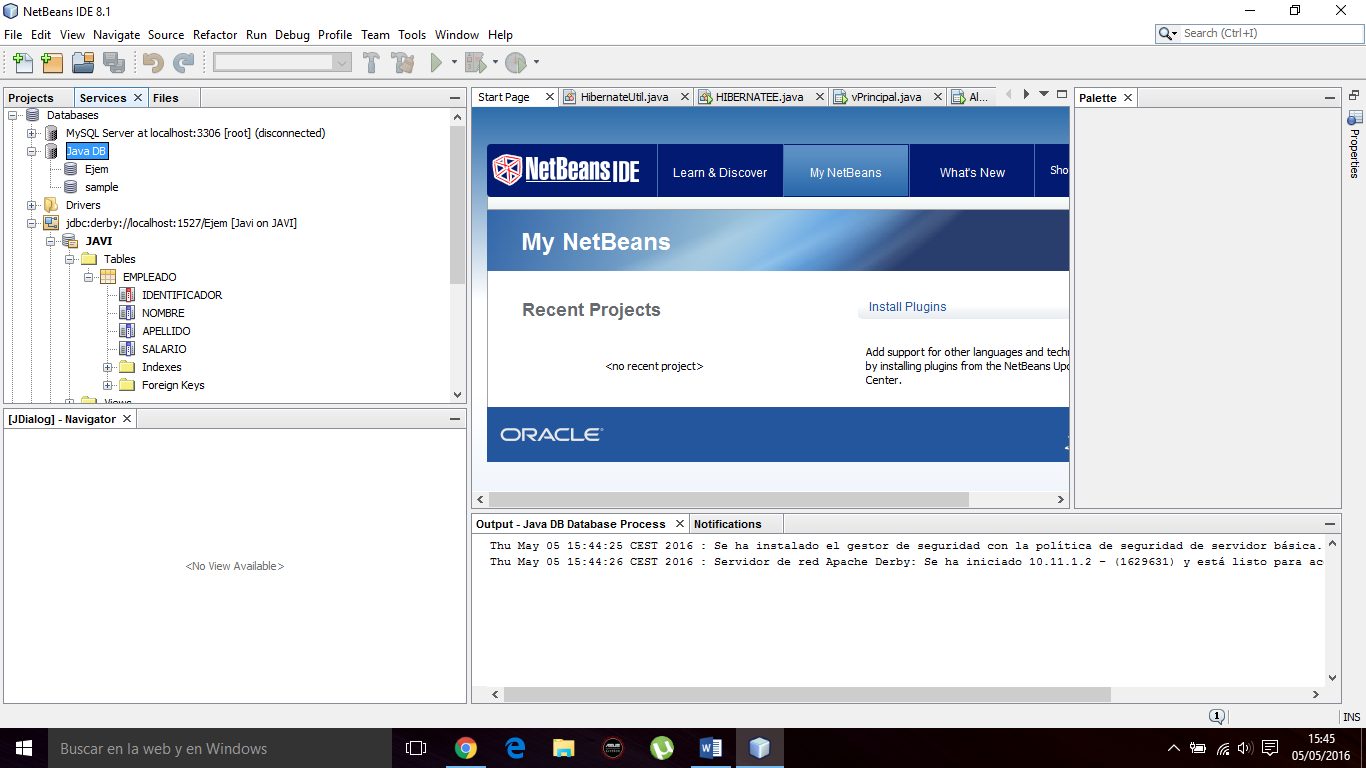


Página Ver Todos Empleados

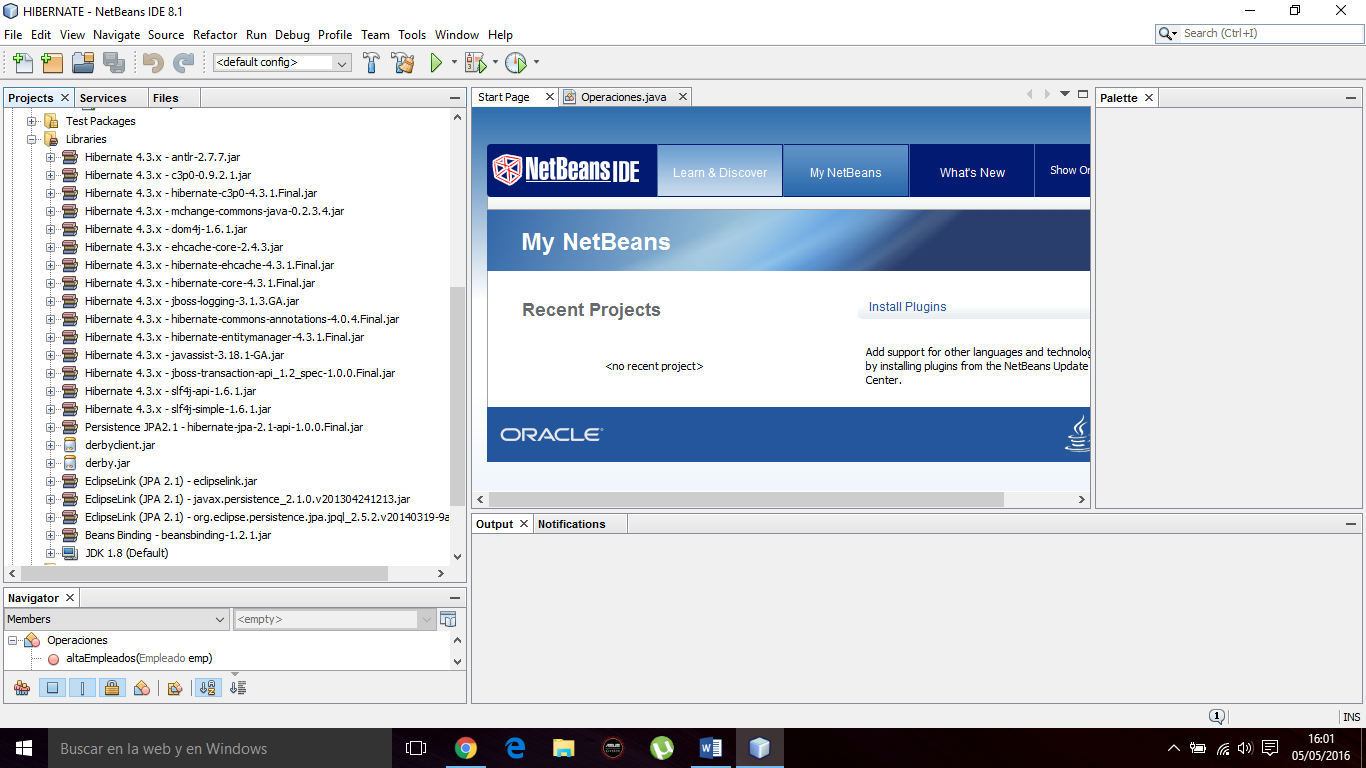
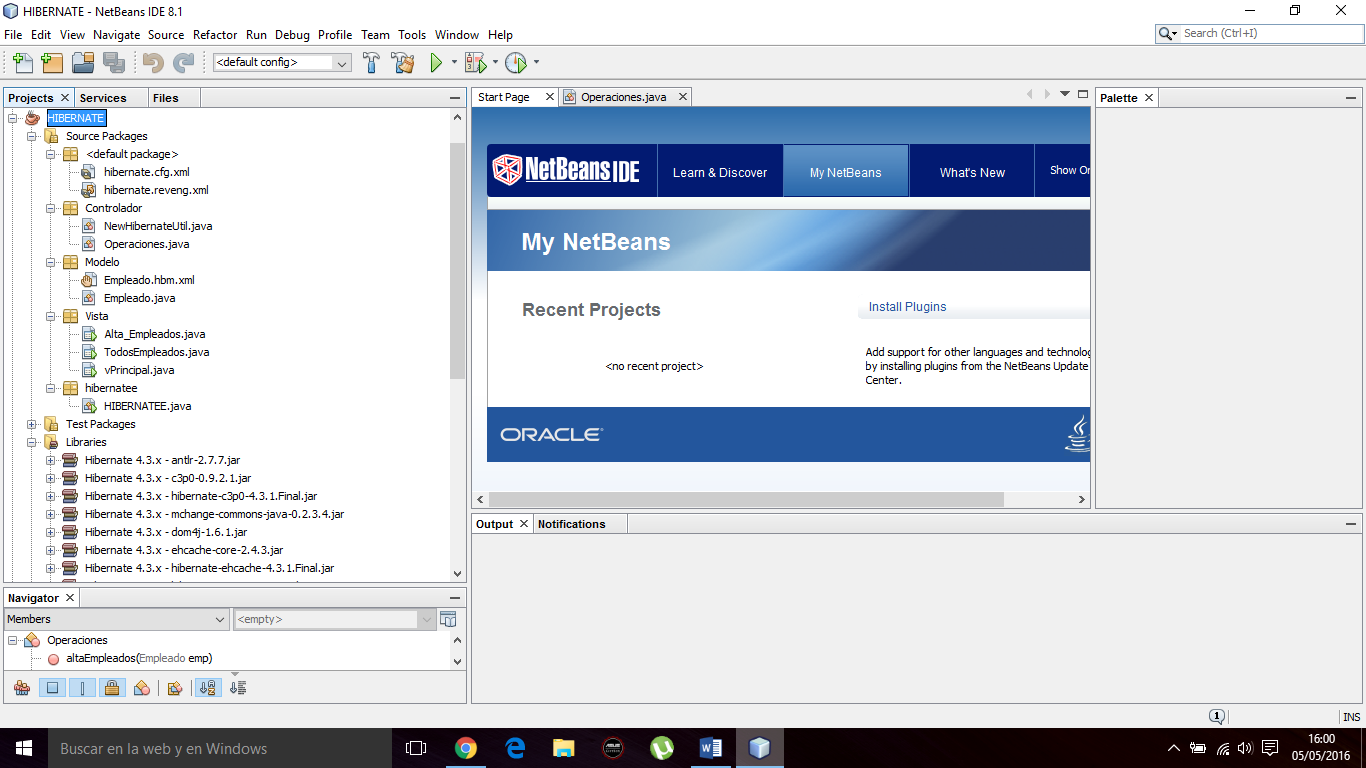
## 4.2 Documentación de construcción

Hay que incluir una descripción de la construcción del prototipo, incluyendo algún extracto de código fuente. No es necesario todo el código. Sólo algún extracto para ver cómo se ha comentado.

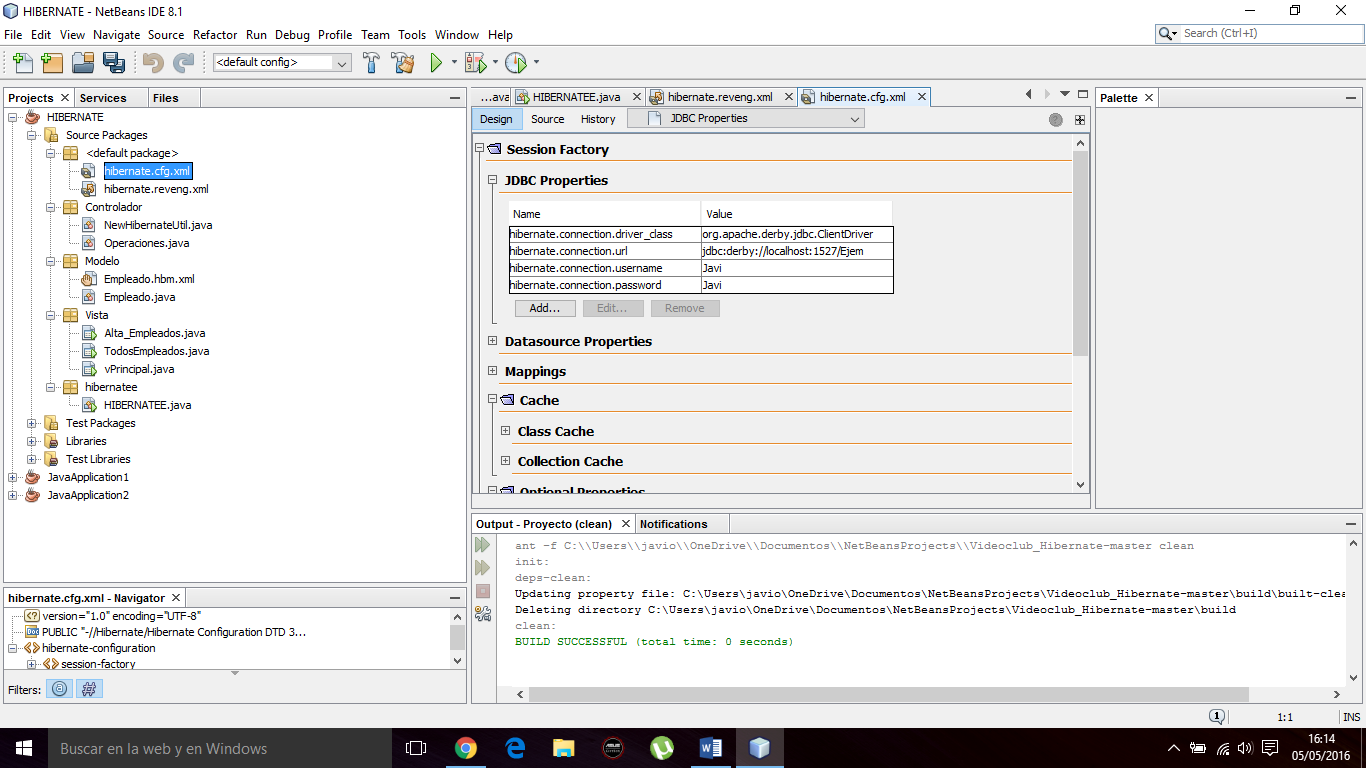
Primero creamos una base de datos, utilizamos la base de datos local de java para dicha realización. Su nombre es Ejem como vemos en el pantallazo y luego como vemos en la parte inferior creamos la tabla empleados con sus propiedades y su clave primaria que es el identificador.



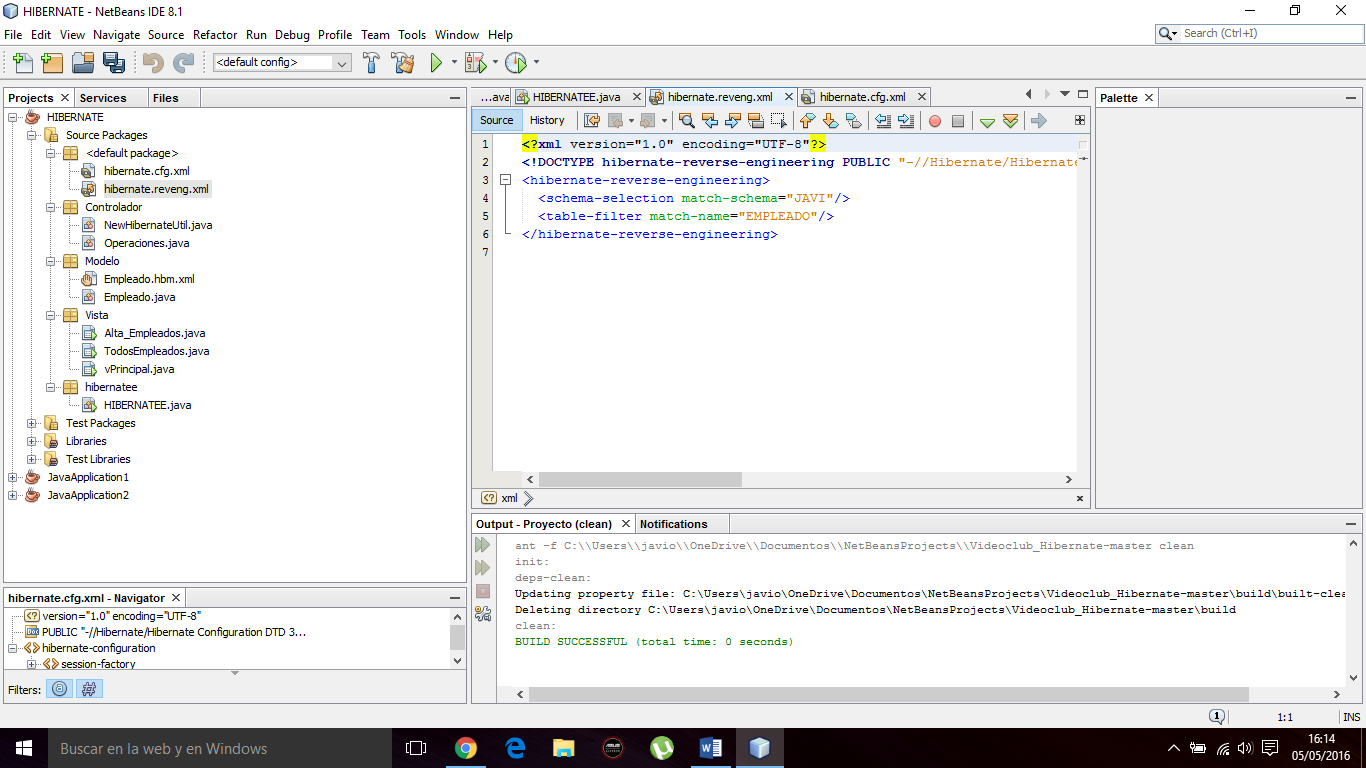
Aquí podemos ver todos los paquetes que incluye nuestro proyecto además de la base de datos que anteriormente hemos visto. Como vemos está dividida en varias carpetas. En la primera están los archivos de configuración de hibernate. Luego otra con los métodos(controlador), Clase objeto (modelo), y vista con las interfaces gráficas y conexiones, por último, la capeta final llamada Hibernate es el comienzo de la aplicación. También podemos ver las librerías que utiliza (Hibernate, JPA2.1,JDK1.8) y la conexión con la base de datos derby.



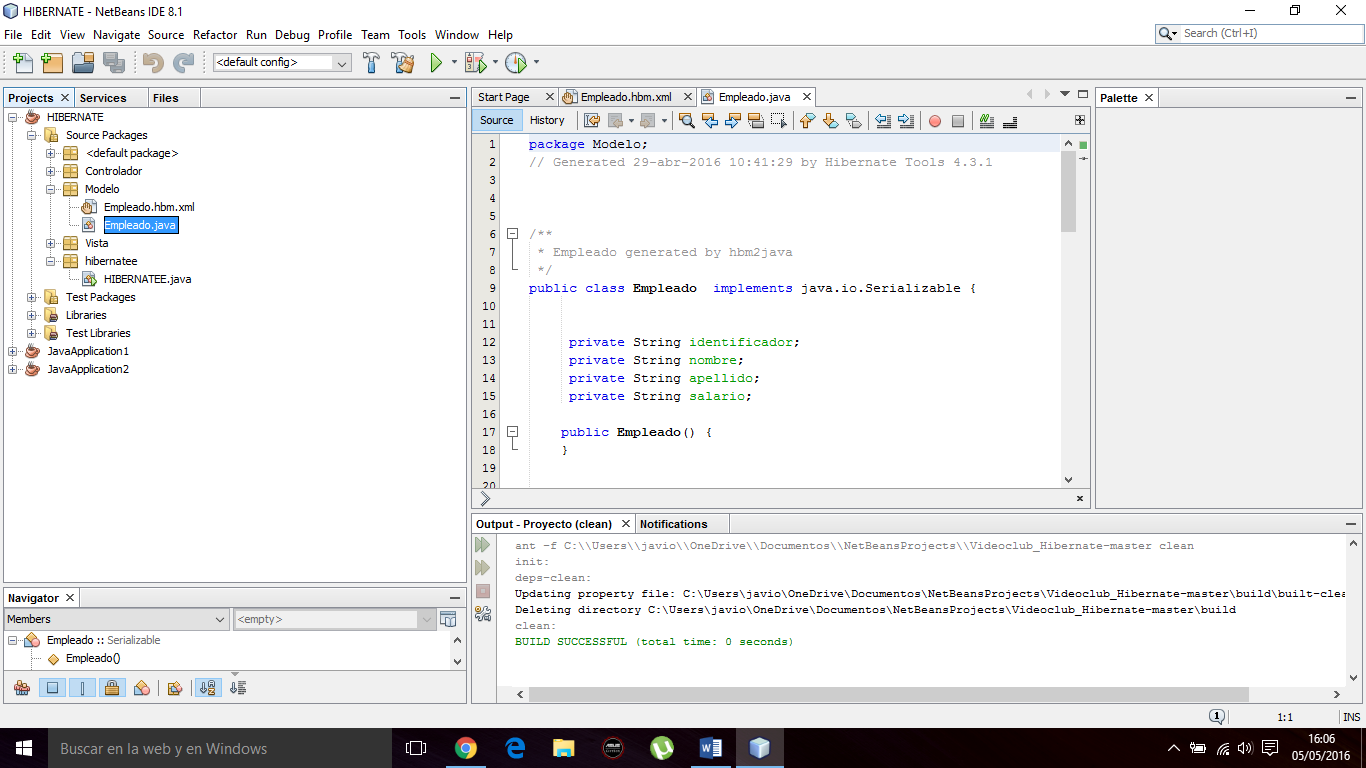
Comenzamos con la aplicación y lo primero que hacemos es configurar hibernate para la conexión con la base de datos, la cual tras unos pasos obtenemos dicho archivo: hibernate.cfg.xml.



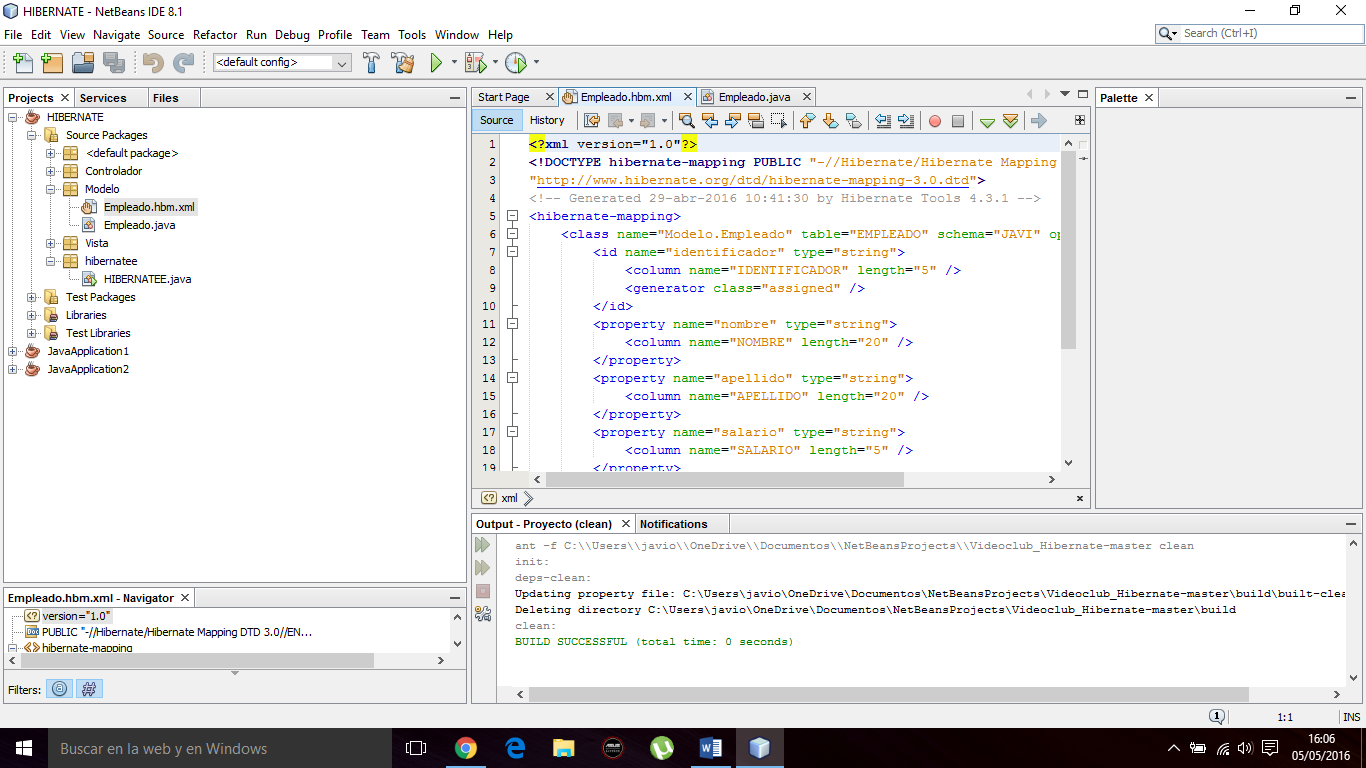
A continuación, realizamos la ingeniería inversa de hibernate, dentro de la cual seleccionamos la tabla que queremos. Obtenemos el archivo hibernate.reveng.xml.



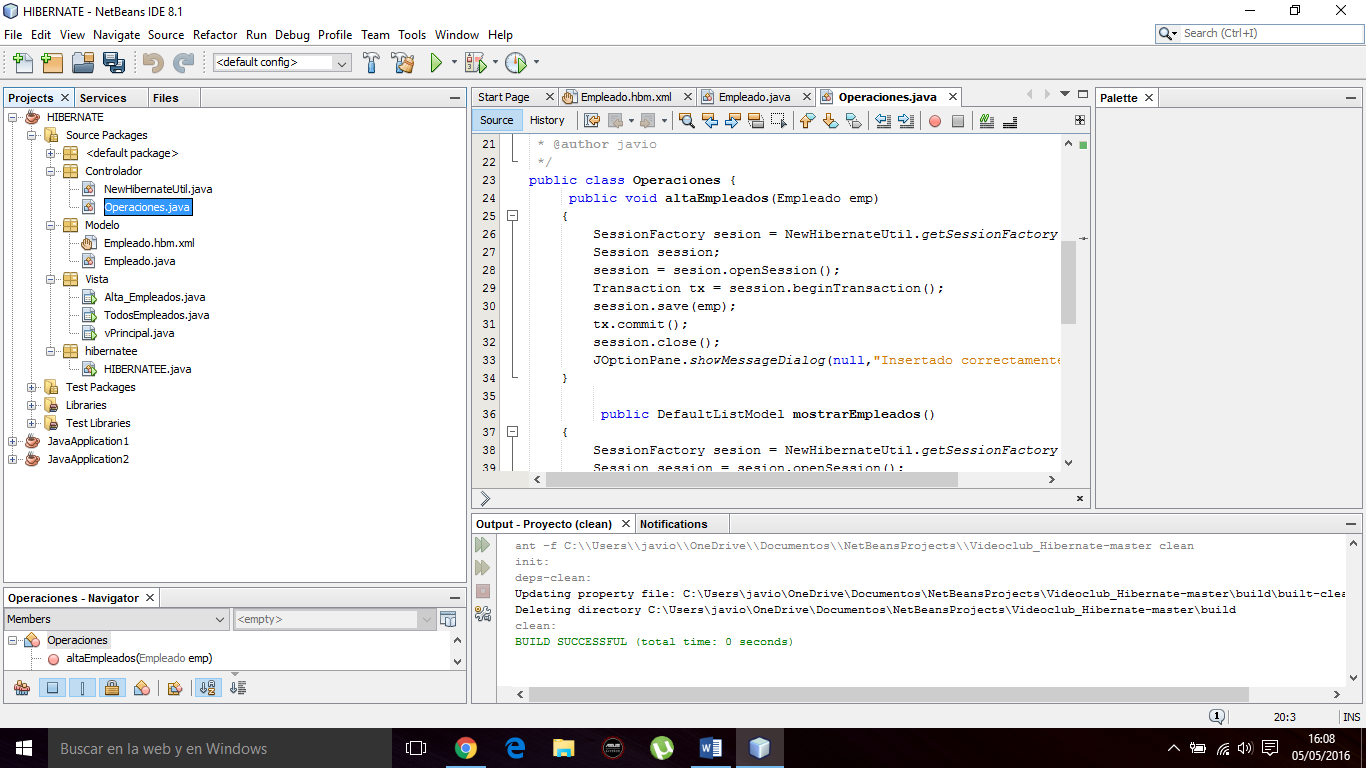
Ahora generamos las clases, a través de Hibernate Mapping Files and POJOs from database, el cual utiliza los archivos anteriormente mencionados y crea una clase normal de java llamada Empleado y…



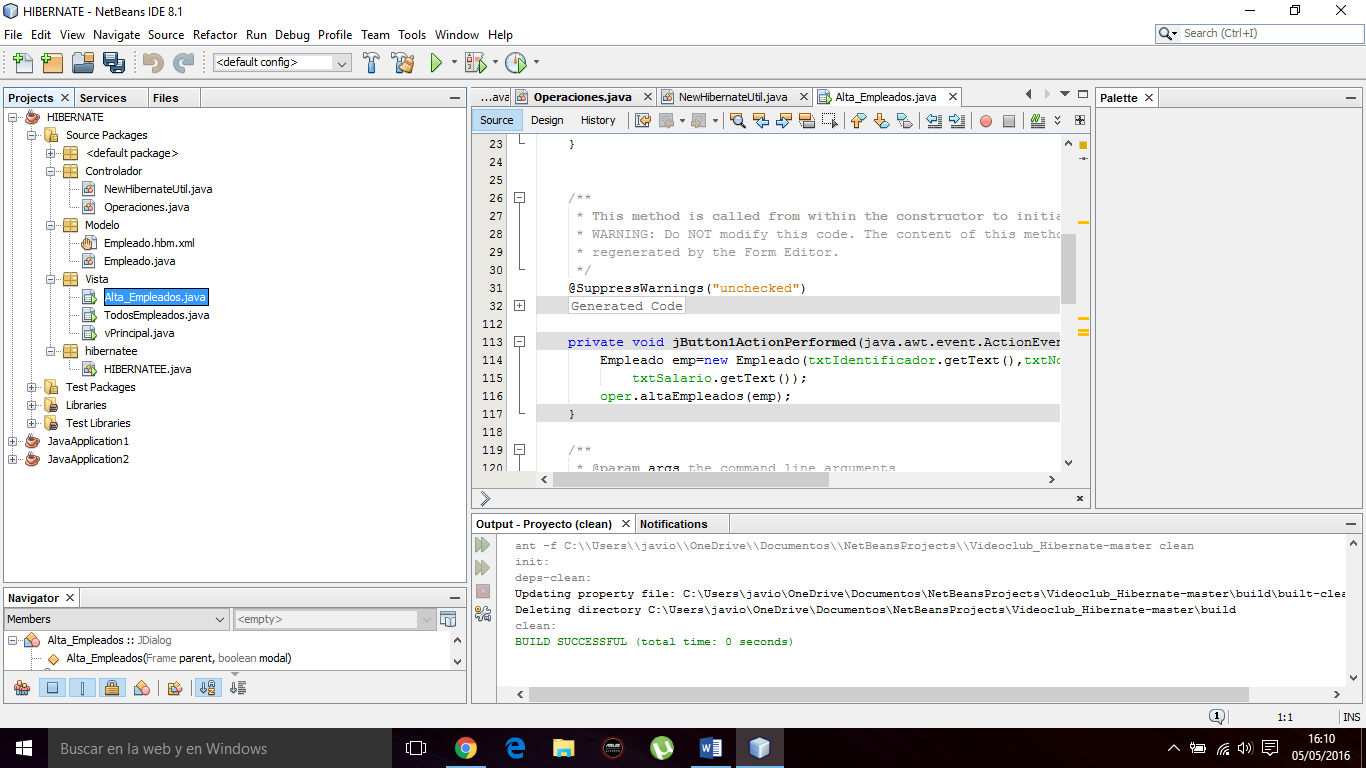
Su configuración en xml llamada Empleado.hbm.xml.



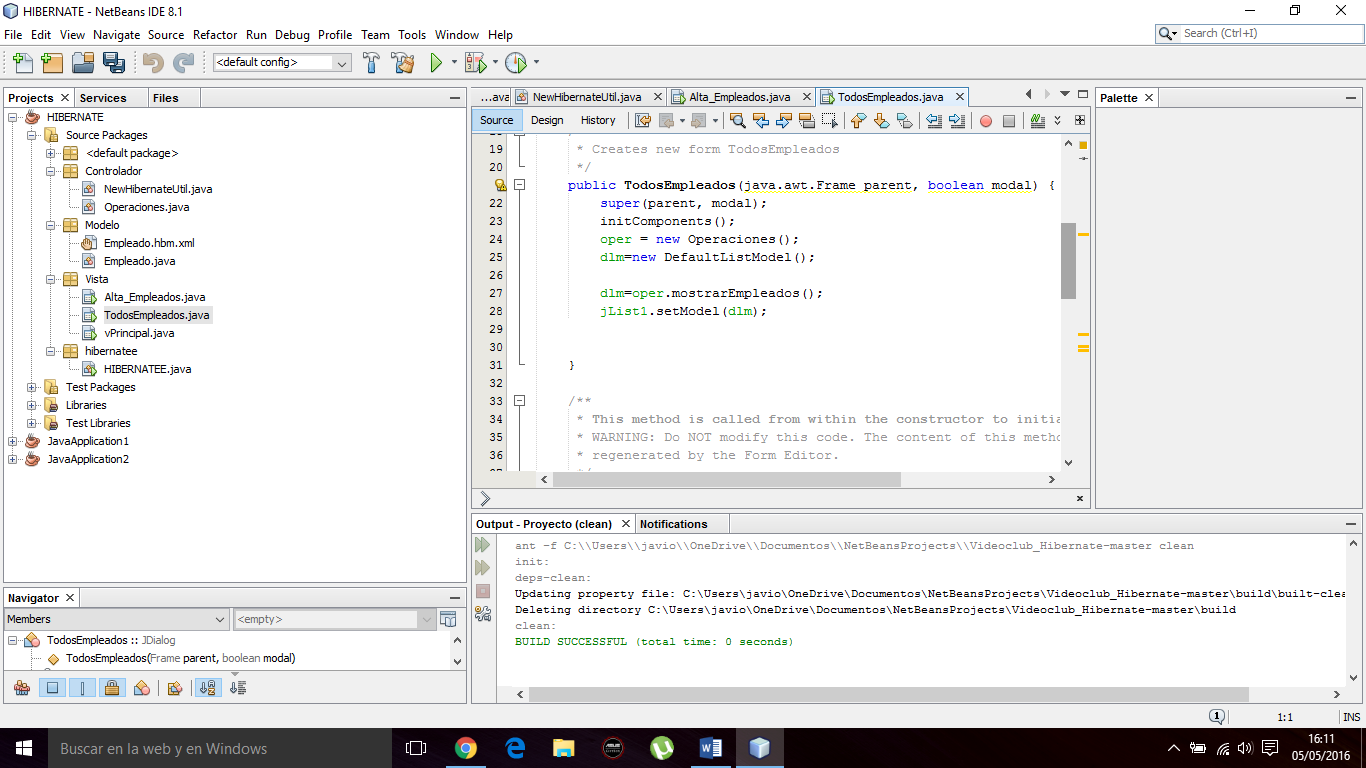
Ahora comenzamos con las operaciones o métodos que va a realizar la aplicación. En este caso tenemos el de dar de alta al empleado y otro que muestras todos los empleados registrados.



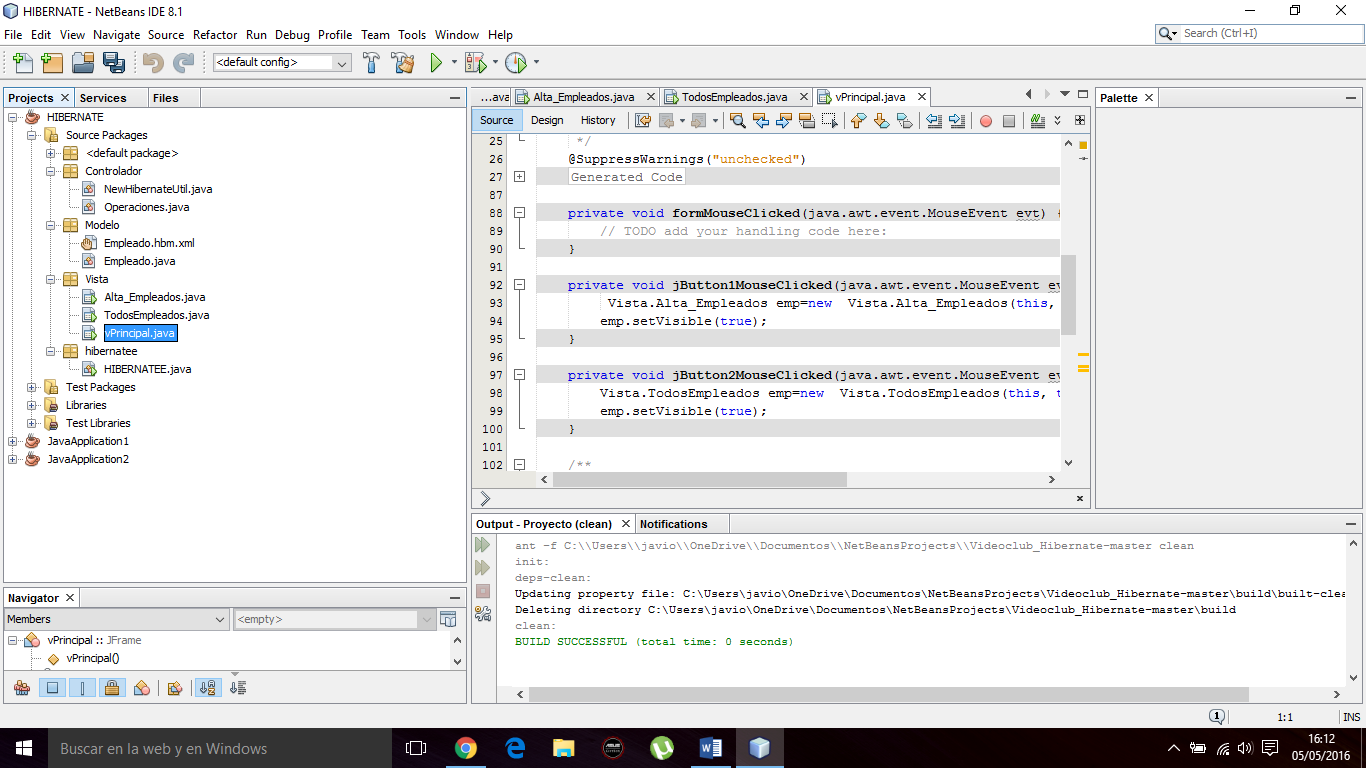
Ahora vamos con las vistas en esta primera es de la de Alta de Empleados dentro de la cual recogemos todos los datos que serán rellenados en el formulario que hemos creado.



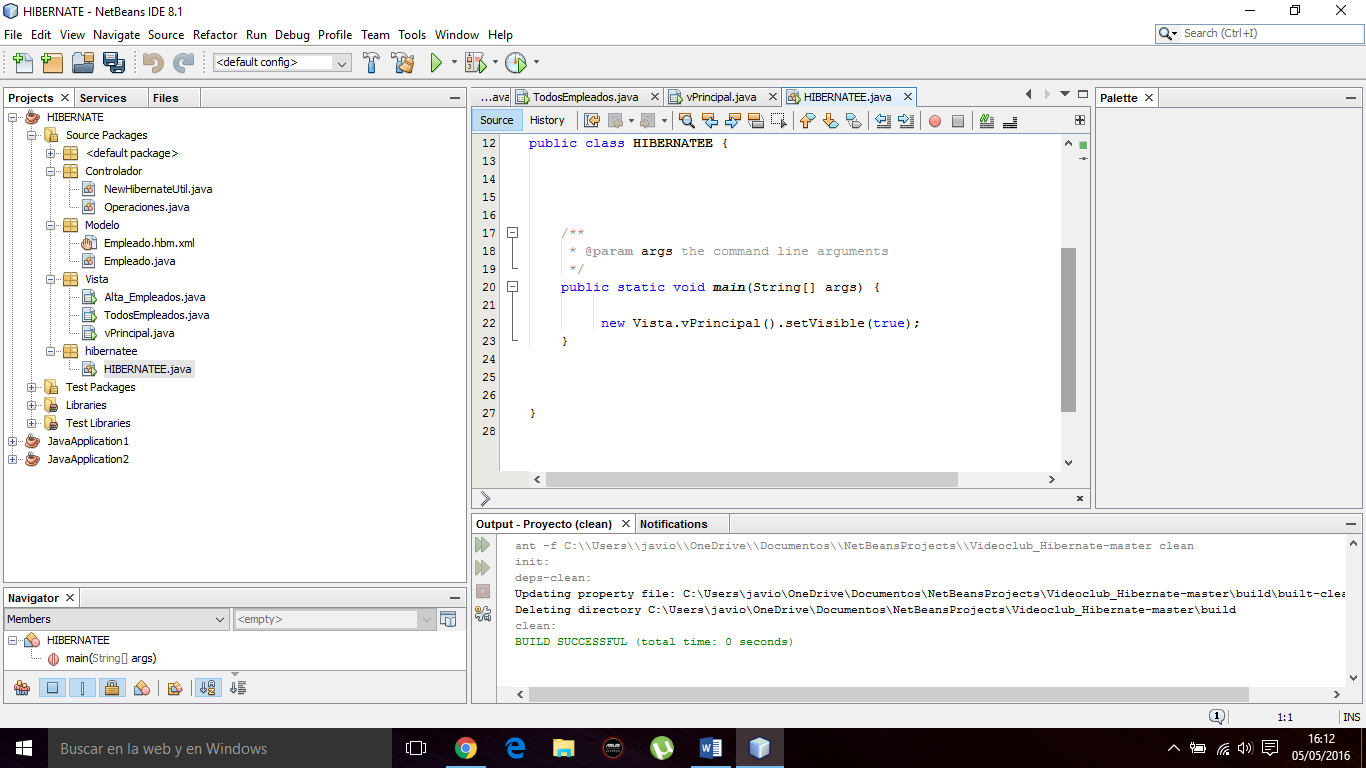
En esta vista mostraremos los datos de los empleados registrados anteriormente.



Por último, esta es la vista de la página principal, la cual observamos el re direccionamiento de los botones a las diversas páginas.



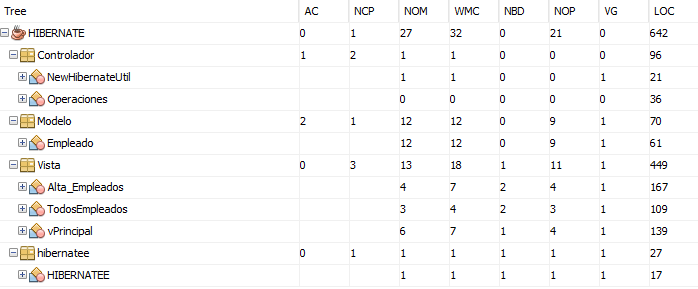
Y aquí estaría el main del programa.



## 4.3 Documentación de pruebas

**Pruebas de caja blanca-Métricas**

Tras completar la aplicación, hemos obtenido unas métricas básicas del propio Netbeans:



LOC: Líneas de Código.

VG: Complejidad ciclomatica

NOP: Número de parámetros

NBD: Profundidad de herencia

NOM: Número de métodos

NCP: Número de clases

AC: Acoplamiento de clases

El proyecto tiene 642 líneas de código, albergando la mayoría de ellas en el paquete Vista, siendo la vista Alta\_Empleados la que más líneas tiene con 167.

En cuanto a la complejidad ciclomatica, todos los valores son o “0” o “1”, que lo califica como un programa simple y sin mucha complejidad.

**Pruebas funcionales**

Estas pruebas se han realizado manualmente, comprobando que la aplicación cumple los requisitos funcionales previamente descritos (RF1, RF2, RF3, RF4 y RF5)

**Pruebas de usabilidad**

Al realizar las pruebas de usabilidad, en las interfaces, hemos detectado que se podían maximizar pero sin adaptar el contenido.

Solución: cambiar la variable resizable a false en todas las interfaces para que no muestren el botón de maximizar.

Esta prueba se ha realizado manualmente.

**Pruebas de caja negra**

A la hora de introducir los datos en los formularios, hemos detectado que no existía un control sobre los datos que se introducían, permitiendo el programa que estos se insertasen vacíos.

Para solucionarlo se ha añadido una sentencia condicional que comprueba que los campos no están vacíos y que el valor numérico del campo Salario no es negativo.

Este tipo de prueba se ha realizado manualmente

## 4.4 Documentación de instalación

Para poder ejecutar el prototipo debe pulsar en el ejecutable que ha sido creado, el cual se abrirá y podrá acceder a la aplicación la cual funcionará una vez esté conectada la base de datos.

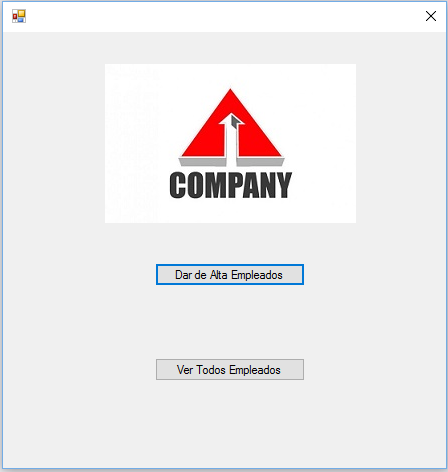
## 4.5 Manual de usuario

La aplicación es muy sencilla cualquier podría saber utilizarla sin la necesidad de anterior explicación. Una vez abres la aplicación se inicia con una página principal la cual consta de dos botones, si pulsamos el primer botón llamado “registrar empleado” accedes a otra página en la cual encontraras un formulario que deberás rellenar con los datos del empleado que quieras registrar en la base de datos y por ultimo para que estos queden guardados debes pulsar el botón “registrar”, si el registro se realizó con éxito te apareceré un mensaje confirmándolo. Si en cambio pulsamos el segundo botón de la página principal llamado “ver empleados” accederemos a otra página en la cual observamos todos los empleados que se encuentran registrados en la base de datos.

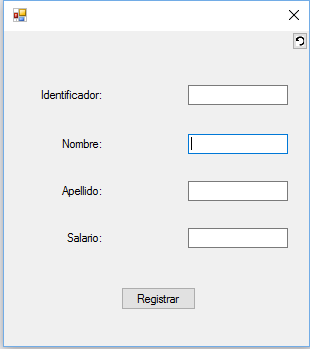
# 5. Proyecto de implementación de un prototipo del sistema utilizando Entity Framework

## 5.1 Documentación de diseño

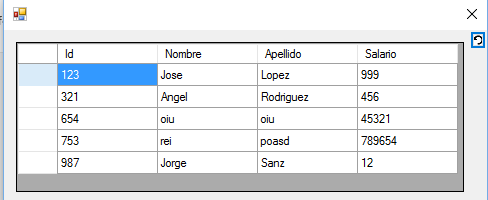
El diseño de la aplicación basada en Entity Framework consiste en una aplicación sobre los empleados de una empresa. Dicha aplicación consta de una página inicial en la cual se incluyen dos botones. Un primer botón llamado “Dar de Alta Empleados” el cuál accederá a otra página, dentro de la cuál tendremos un formulario que rellenar los datos del empleado, el registro se confirmará con un botón llamado “Registrar”, con lo que los datos se guardaran en la base de datos creada anteriormente a través de Entity Framework. El segundo botón de la página inicial llamado “Ver Todos Empleados”, el cuál accederá a otra página que nos mostrará todos los empleados anteriormente registrados.



*Página principal*



*Página añadir empleado*



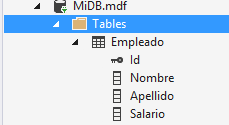
*Página de empleados*

## 5.2 Documentación de construcción

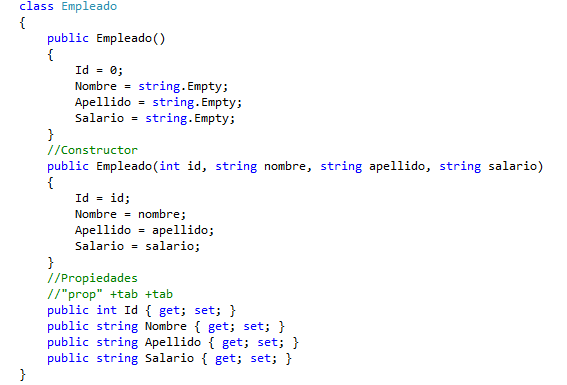
Comenzamos creando la base de datos, para ello utilizamos SQL Server integrado en visual studio, agregamos una nueva base de datos, en nuestro caso la hemos llamado “MiDB”, una vez la tengamos creadas, solo tenemos que añadir las tablas que deseemos. Podemos hacerlo de dos formas, una, utilizando el SQL, es decir añadir el código “ CREATE TABLE Empleados ID INT PRIMARY KEY, NOMBRE VARCHAR(10), APELLIDO VARCHAR(10), SALARIO INT”, o de la forma más visual, que es agregando tabla a la base de datos por medio de la interfaz:

# 

En la tabla añadimos el nombre de las entidades y ponemos el tipo correspondiente. Una vez lo tengamos presionamos en “Update” y ya estará creada nuestra tabla:



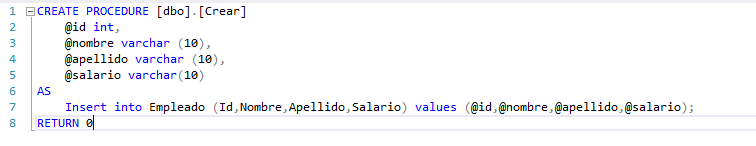
Ahora comenzamos el diseño de nuestra aplicación. En nuestro proyecto añadimos la carpeta “model”, esta carpeta introduciremos la clase empleado. En ella iniciaremos los valores de nuestra base de datos a 0 en caso de los enteros y a vacío en caso de los VARCHAR. Crearemos los métodos get y set para acceder a los datos y cargarlos



También añadiremos la carpeta “controller”, en ella crearemos los métodos para acceder a la base de datos y también el procedimiento para añadir los datos a la base de datos. Este procedimiento lo creamos en la base de datos y después se invoca en este apartado:



También creamos el procedimiento:

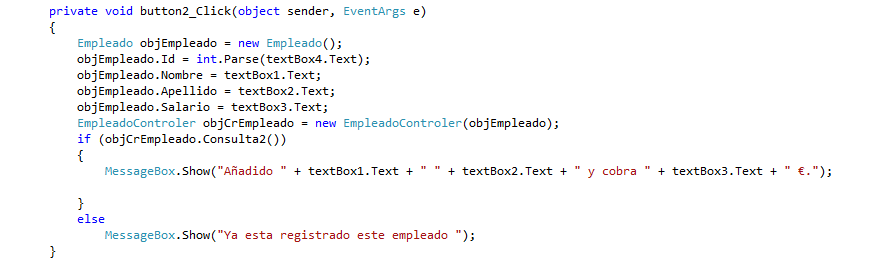


Una vez tenemos esto, pasamos a las vistas. Para ello hemos utilizado Windows forms.

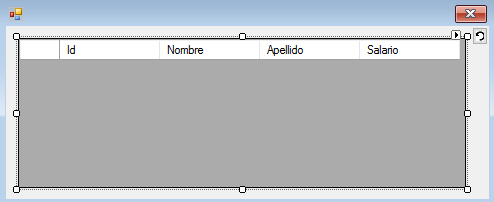
En crear empleado establecemos 4 “textBox” y 4 “label’s” que identifican cada uno:

# 

Este es su código, que toma los valores de los textBox y ejecuta el procedimiento de crear empleado, que almacena los valores en la Base de datos.



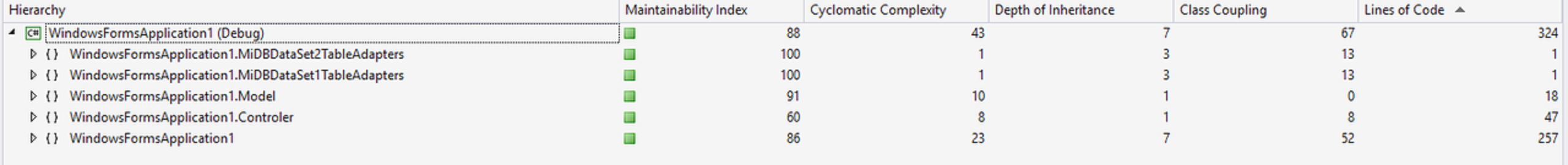
Continuamos con la otra vista, la de mostrar empleados. En ella ponemos un “datagridview” e importamos la base de datos con las columnas que queremos mostrar, así cada vez que entremos en esta sección vemos el contenido de la base de datos.



## 5.3 Documentación de pruebas

**Pruebas de caja blanca- Métricas:**

Las métricas obtenidas a continuación son las que proporciona Visual Studio:



De izquierda a derecha, las métricas obtenidas son: Índice de mantenimiento, complejidad ciclomatica, profundidad de herencia, acoplamiento de clase y líneas de código.

Como se puede observar, las líneas de código son 324, acumulando 257 de ellas en las interfaces gráficas.

En cuanto a la complejidad ciclomatica, es bastante elevada, 43, clasificando el programa como complejo y de alto riesgo.

**Pruebas funcionales**

Estas pruebas se han realizado manualmente, comprobando que la aplicación cumple los requisitos funcionales previamente descritos (RF1, RF2, RF3, RF4 y RF5)

**Pruebas de Usabilidad**

Al realizar las pruebas de usabilidad, no hemos detectado ningún fallo o error, por lo que nos ha servido para confirmar su facilidad de uso y eficiencia. Cumpliendo así los requisitos no funcionales.

Esta prueba se ha realizado manualmente, comprobando su facilidad de manejo y los tiempos de reacción de la aplicación.

**Pruebas de caja negra**

A la hora de introducir los datos en los formularios, hemos detectado que no existía un control sobre los datos que se introducían, permitiendo el programa que estos se insertasen vacíos.

Para solucionarlo se ha añadido una sentencia condicional que comprueba que los campos no están vacíos y que el valor numérico del campo Salario no es negativo.

Esta prueba se ha realizado manualmente.

# 6. Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## 6.1 Evaluación de los criterios en la implementación usando la tecnología Hibernate

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Dificultad de aprendizaje | 7-8 |
| Número de líneas | 627 LOC |
| Tiempo de Consulta | 40 milisegundos |
| Base de Datos | Se ha utilizado la base de datos local de Netbeans |
| Peso Aplicación | 50.1 MB |
| Look and Feel | La apariencia de las interfaces son muy clásicas y “aburridas” |
| Uso de CPU | 12.8% |
| Uso de memoria | 20MB |
| Complejidad ciclomatica | 9 |

## 6.2 Evaluación de los criterios en la implementación usando la tecnología Entity Framework

| **CRITERIO** | **EVALUACIÓN** |
| --- | --- |
| Dificultad de aprendizaje | 3-4 |
| Número de líneas | 324 LOC |
| Tiempo de Consulta | 31 milisegundos |
| Base de Datos | SQL server de Visual Studio |
| Peso Aplicación | 33.1MB |
| Look and feel | Tiene una interfaz clásica |
| Uso de CPU | 11.25% |
| Uso de Memoria | 21 MB |
| Complejidad ciclomatica | 43 |

# 7. Comparación de la implementación de las tecnologías

| **CRITERIOS** | **Hibernate** | **Entity Framework** | **COMENTARIOS** |
| --- | --- | --- | --- |
| Dificultad de aprendizaje | 7-8 | 3-4 | Aquí la diferencia está muy clara. Entity tiene una curva de aprendizaje más suave que Hibernate. Aprendiendo más rápido a usarlo y siendo mucho más sencillo. |
| Número de líneas | 627 LOC | 324LOC | Hibernate genera casi el doble de líneas de código ya que requiere más archivos que entity para su funcionamiento. |
| Tiempo de consulta | 40 milisegundos | 31 milisegundos | Al ser tiempos tan pequeños, la diferencia es nimia, por lo que se podría decir que ambos son igual de rápidos. |
| Base de datos | Apache Derby | SQL server | Ambas están empotradas en los entornos de desarrollo usados, por lo que la conexión ha sido fácil de realizar |
| Peso Aplicación | 50.1MB | 33.21MB | Al tener casi el doble de líneas de código que Entity, es lógico que la aplicación basada en Hibernate pese casi el doble, ya que incluye más ficheros. |
| Look and feel | Clásico | Clásico | En cuanto al Look and Feel ambas aplicaciones se han desarrollado de manera similar. Lo único que cabe resaltar es que con Visual Studio se podía dar más “estilo” a las interfaces que con Netbeans. |
| Uso de CPU | 12.8% | 11.25% | El uso de CPU es muy similar en ambas aplicaciones. |
| Uso de memoria | 20MB | 21MB | Al igual que pasa con el uso de CPU, el uso de memoria es casi idéntico en las dos aplicaciones. |
| Complejidad ciclomatica | 9 | 43 | En este apartado es donde nos hemos llevado la mayor sorpresa, al comprobar que la aplicación realizada con Hibernate, que cuenta con más líneas de código, obtiene una mejor puntuación de complejidad ciclomática, clasificándola como una aplicación simple y sin riesgos.  Por el contrario, para la aplicación con Entity, la complejidad ciclomatica se dispara, clasificándola como compleja y con mucho riesgo. |

# 8. Conclusiones

Tras analizar los resultados obtenidos de cada aplicación, y después de compáralos en el apartado 7 y teniendo en cuenta nuestras impresiones a la hora de realizarlas, hemos llegado a las siguientes conclusiones:

En cuanto al aprendizaje, nos ha resultado mucho más fácil dominar Entity Framework que Hibernate, este último tiene un aprendizaje más pesado y menos dinámico que Entity Framework.

Las líneas de código no las vamos a contar como un factor determinante, y que la mayoría de ellas han sido generadas automáticamente por los Frameworks.

A la hora de decidir cuál de las 2 herramientas es mejor, nuestra respuesta es: depende. Nos explicamos: En cuanto al diseño de la interfaz y facilidad de manejo, nos quedamos con Entity Framework, pero en cuanto a fiabilidad y robustez nuestra elección es Hibernate, ya que nos ha dado menos problemas técnicos y ha obtenido mejor nota que Entity en complejidad ciclomatica.