Contenido

[1. Autores del trabajo, planificación y entrega 2](#_Toc445388848)

[1.1 Autores 2](#_Toc445388849)

[1.2 Planificación 2](#_Toc445388850)

[1.3 Entrega 2](#_Toc445388851)

[2. Descripción de las tecnologías 3](#_Toc445388852)

[2.1 Descripción de la tecnología 1 3](#_Toc445388853)

[2.2 Descripción de la tecnología 2 3](#_Toc445388854)

[3. Criterios de comparación 4](#_Toc445388855)

[3.1 Categoría A: Nombre 4](#_Toc445388856)

[3.1.1 Criterio A.1: Nombre 4](#_Toc445388857)

[3.1.2 Criterio A.2: Nombre 5](#_Toc445388858)

[3.1.n Criterio A.n: Nombre 5](#_Toc445388859)

[3.2 Categoría B: Nombre 5](#_Toc445388860)

[3.2.1 Criterio B.1: Nombre 5](#_Toc445388861)

[3.2.2 Criterio B.2: Nombre 5](#_Toc445388862)

[3.2.n Criterio B.n: Nombre 5](#_Toc445388863)

[3.3 Categoría Z: Nombre 5](#_Toc445388864)

[3.3.1 Criterio Z.1: Nombre 5](#_Toc445388865)

[3.3.2 Criterio Z.2: Nombre 5](#_Toc445388866)

[3.3.n Criterio Z.n: Nombre 5](#_Toc445388867)

[4. Evaluación de los criterios por tecnología 6](#_Toc445388868)

[4.1 Evaluación de los criterios para la tecnología 1 6](#_Toc445388869)

[4.2 Evaluación de los criterios para la tecnología 2 6](#_Toc445388870)

[5. Comparación de las tecnologías 7](#_Toc445388871)

[6. Recomendaciones 9](#_Toc445388872)

[6.1 Situación 1 9](#_Toc445388873)

[6.1.1 Descripción de la situación 9](#_Toc445388874)

[6.1.2 Recomendación de tecnología a utilizar 9](#_Toc445388875)

[6.2 Situación 2 9](#_Toc445388876)

[6.2.1 Descripción de la situación 9](#_Toc445388877)

[6.2.2 Recomendación de tecnología a utilizar 9](#_Toc445388878)

# 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/5381d5471d9b6f1fd65ae42ad9a97666b9e701480398d76f7f50dd40974ae8d8>

## 1.3 Entrega

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

<https://github.com/JaviOblare/TG2.git>

En dicho repositorio debe encontrarse, al menos los siguientes archivos en la rama máster:

* Trabajo terminado: del trabajo terminado con el nombre TG2\_final.docx
* Presentación del trabajo: TG2\_final.pptx

Dichos archivos serán los que se tendrán en cuenta para la calificación del trabajo.

# 2. Descripción de las tecnologías ORM (Hibernate & Entity Framework)

En los siguientes apartados se debe describir brevemente cada tecnología a comparar.

Se pueden incluir imágenes copiadas y pegadas desde diversas fuentes de información, siempre que debajo de cada una se indique la fuente (al menos la URL).

## 2.1 Descripción de Hibernate

**¿QUÉ ES HIBERNATE?**

Para introducirnos en el mundo de Hibernate primero tomaremos una definición:

“Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre deNHibernate) 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 losbeansde las entidades que permiten establecer estas relaciones. Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL”.

En otras palabras, Hibernate es un Framework que agiliza la relación entre la aplicación y la base de datos. Para poder aprender a utilizarlo es necesario contar con los conocimientos básicos de base de datos y SQL  así como manejar el lenguaje Java.

**HISTORIA**

Hibernate fue una iniciativa de un grupo de desarrolladores dispersos alrededor del mundo conducidos por Gavin King.  
  
Hibernate se inició en 2001 por Gavin King con colegas de Cirrus Tecnologías como una alternativa al uso de beans de entidad estilo EJB2. Su objetivo original era ofrecer una mejor capacidad de persistencia de los ofrecidos por EJB2 simplificando las complejidades y complementa las características faltantes.  
  
A principios de 2003, el equipo de desarrollo Hibernate comenzó a dar comunicados de Hibernate2, que ofrecía muchas mejoras significativas con respecto a la primera versión.  
  
Tiempo después, JBoss Inc. (ahora parte de Red Hat) contrató a los principales desarrolladores de Hibernate y trabajó con ellos en brindar soporte al proyecto.  
  
En 2005, Hibernate versión 3.0 fue lanzado. Las características claves incluyen una nueva arquitectura Interceptor/Callback, filtros definidos por el usuario, y JDK 5.0 Anotaciones(Java metadatos de función).

En 2005 tambien fue lanzado **NHibernate** es la conversión de [Hibernate](https://es.wikipedia.org/wiki/Hibernate" \o "Hibernate) de lenguaje [Java](https://es.wikipedia.org/wiki/Java_(Sun)) a [C#](https://es.wikipedia.org/wiki/C_Sharp) para su integración en la plataforma .NET.  Su última versión estable es la 4.0.3

A partir de 2010, Hibernate 3(versión 3.5.0 o superior) era una aplicación certificada en la Java Persistence API 2.0 especificación a través de un contenedor para el módulo Core que proporcionaba la conformidad con el JSR 317 estándar.  
  
En diciembre de 2011, Hibernate Core 4.0.0 fué puest en libertad. Esto inluye nuevas características como el soporte multi-alquiler, la introducción de ServiceRegistry(un cambio importante en la forma de hibernación construye y gestiona "servicios"), una mejor apertura de sesión de SessionFactory, mejora la integración a través de org.hibernate.integrator.spi.integrator y detección automática, internacionalización de códigos de soporte y mensajes de registro, una división más clara entre el API, SPI y clases de implementación.  
  
En diciembre de 2012, Hibernate ORM 4.1.9 Se lanzó al mercado.

En 2012 también se inició el 5 de Hibernate. Contendrá JPA 2.1

La actual versión de Hibernate disponible es la 5.1.0 y su última versión estable es la 4.3.1.

**CARACTERISTICAS:**

Estas son solo algunas de las características que nos ofrece Hibernate

* Herramienta de Mapeo objeto relacional (ORM) para Java.
* Facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante ficheros declarativos (XML) o anotaciones en los beans
* Software libre
* Busca solucionar el problema entre el modelo de memoria de una aplicación (orientado a objetos) y el usado en base de datos (modelo relacional).
* Hibernate permite a la aplicación manipular los datos de la base operando sobre los objetos, que tiene las características de la POO.
* Hibernate convertirá los datos entre los tipos utilizados en Java y los definidos por SQL.
* Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias.
* Mantiene la portabilidad entre los motores de base de datos, con un ligero incremente en el tiempo de ejecución.
* Es flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente.
* También puede crear una base de datos a partir de la información disponible.
* Además, ofrece un lenguaje de consulta de datos llamado HQL, además de un API para construir las consultas programáticamente (criteria).
* Puede utilizarse en aplicaciones Java independientes (o standalones) o bien en aplicaciones Java EE, mediante el componente Hibernate Annotations.

## 2.2 Descripción de Entity Framework

# ¿Qué es Entity Framework?

Es un conjunto de tecnologías de ADO.NET que permiten desarrollar aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones se han estado enfrentando a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar datos. Los datos pueden abarcar varios sistemas de almacenamiento, con protocolos propios, incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicas sin tener que preocuparse por las tablas y columnas de la base de datos. Los desarrolladores pueden trabajar a un mayor nivel de abstracción cuanto tratan datos y pueden crear y mantener aplicaciones con menos código que en las aplicaciones tradicionales.

# Historia:

El líder del equipo de desarrollo del .NET Framework fue Anders Hejsberg, un ingeniero de software danés muy conocido en el medio y actual Arquitecto Jefe de C# en Microsoft. Cuenta la leyenda que corría el año 1996 y Anders pertenecía a Borland. Entre sus principales aportes en esta empresa fue liderar los proyectos de desarrollo de Turbo Pascal y Delphi. Fue entonces cuando Microsoft lo tuvo como objetivo primordial: primero con una propuesta muy atractiva y luego duplicando la contra-propuesta económica de Borland. No se podía competir con Microsoft en ese entonces.

El objetivo inicial era una implementación de Java de la propia Microsoft, que tenía que integrarse en Visual Studio. Este nuevo lenguaje se llamó en ese momento J++, pero luego se mejoró la idea y el lenguaje terminó por convertirse en lo que hoy conocemos como C#. Además, el proyecto se perfiló mejor y se convirtió en nuestra plataforma de desarrollo favorita. Fue así que la versión 1.0 fue lanzada oficialmente en enero del 2002.

De forma paralela al crecimiento del .NET Framework, también ha evolucionado el entorno de desarrollo que acompaña esta tecnología: [Visual Studio](http://www.visualstudio.com/es-es). Es justo decir que Microsoft ha alcanzado un producto de calidad con este IDE, que acompaña de forma perfecta a la plataforma. En los últimos años hay un impulso muy fuerte por parte de Microsoft en las tecnologías .NET, por lo que a los desarrolladores que tratamos de seguirle el paso nos tiene bastantes exhaustos, pero satisfechos. Entre las principales tecnologías tenemos: Windows Forms, ASP .NET, WCF, WPF, Entity Framework, ASP .NET MVC, ASP .NET WebPages, WebApi, SignalR, Windows Azure, etc.

Finalmente se puede decir que ahora el .NET Framework se ha convertido en un [gigantesco ecosistema](http://www.microsoft.com/en-us/download/details.aspx?id=39952), con tecnologías de calidad, maduras y consolidadas. Además cuenta con una comunidad de desarrolladores muy activa y con el respaldo de una empresa como Microsoft, que no es poca cosa: La división de desarrollo es, según mi punto de vista, la más exitosa de la empresa.

A continuación tenemos una tabla con las versiones de EF , su fecha de lanzamiento y característica propia:

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Características |
| V1 | 2008 | Incluida en .Net Framework 3.5 SP1 y visual studio 2008 |
| V2 | 2010 | Llamada EFv4, realizada como parte de .Net 4.0 resuelve muchas críticas y carencias de la anterior versión. |
| V3 | 2011 | EFv4.1 Tiene apoyo de Code First. |
| V4.1 | 2011 | EFv4.1 Update 1 incluye corrección de errores y nuevos soportes. |
| V4.3.1 | 2012 | Incluye soporte a la migración |
| V5 | 2012 | Incluido en .Net Framework 4.5. |
| V6 | 2013 | Licenciado bajo Apache v2, es de código abierto. |
| V7 | - | Está en desarrollo |

# Características:

* Por medio de Entity Data Model (EDM) es posible especificar un diagrama de diseño, usado para generar las clases programables que usa el código de la aplicación.
* Con Object Services se puede trabajar con las clases de “Common Language Runtime” generadas a partir del modelo conceptual.
* Proporciona compatibilidad de infraestructura.
* Compatibilidad con consultas SQL.
* LINQ to Entities proporciona compatibilidad con Language-Integrated Query para consultar entidades.
* El componente de metadatos ADO.NET administra los metadatos en cuanto a las necesidades de tiempo de ejecución y tiempo de diseño de EF.
* Incluye herramientas en evolución que generan asignaciones y clases parciales que representan las entidades en el modelo conceptual.
* Posee un diagrama de arquitectura que visiona la relación de las diversas interfaces .

# 3. Criterios de comparación

En los sub-apartados de este apartado se deben indicar cada uno de los criterios (también llamados factores, propiedades, características, indicadores, etc.), que se usarán en la comparación.

Los criterios deben organizarse en categorías (al menos 3 categorías). El número de criterios totales no puede ser inferior a 20.

Las categorías dependerán del tipo de tecnología, pueden ser categorías como “General”, “Utilidades”, “Rendimiento”, etc.

Los criterios a definir en cada categoría también dependerán del tipo de tecnologías a comparar. En el siguiente apartado hay algunos ejemplos.

## 3.1 Categoría A: GENERAL

3.1.1 Precio de la herramienta

Nombre del criterio: Precio.

Descripción: Valor por el cual se puede adquirir el software para poder utilizarlo.

Tipo de valor: Texto libre.

### 3.1.2 Criterio A.2: Nombre

### 3.1.n Criterio A.n: Nombre

## 3.2 Categoría B: REQUERIMIENTOS

3.2.1 Supported Database Management Systems

Nombre del criterio: Sistemas gestores soportados

Descripción: Sistema gestores de bases de datos que son soportados por la herramienta.

Tipo de valor: Texto libre.

## 

## 3.3 Categoría C: RENDIMIENTO/EFICIENCIA

3.3.1 TTL(Time to Load)

Nombre del criterio:TTL

Descripción: Este es el tiempo entre el comienzo del índice de referencia y la ejecución de la primera consulta.

Tipo de valor: Numérico

3.3.2 SQ(Symple Query)

Nombre del criterio: Consulta Sencilla

Descripción: Tiempo medio que tarda en realizar una consulta sencilla.

Tipo de valor: Numérico

3.3.3 CQ(Complex Query)

Nombre del criterio: Consulta Compleja

Descripción: Tiempo medio que tarda en realizar una consulta compleja, incluye projections, joins, grouping and sorting.

Tipo de valor: Numérico

3.3.4 EL (eager loading)

Nombre del criterio: una consulta con la carga ansiosa

Descripción: rendimiento ante una consulta con una carga ansiosa de datos.

Tipo de valor: Numérico

3.3.5 CUD (creation, update and delete)

Nombre del criterio: creación, actualización y borrado de operaciones.

Descripción: rendimiento de la ejecución de las operaciones básicas como insertar, actualizar y borrar.

Tipo de valor: Numérico

3.3.6 CPU

Nombre del criterio: el tiempo de uso de la CPU

Descripción: es la cantidad de tiempo en que la unidad central de proceso es usada por las instrucciones del programa.

Tipo de valor: Numérico

3.3.7 RAM

Nombre del criterio: uso de memoria media

Descripción: Porcentaje medio de memoria RAM utilizada por el software durante su ejecución.

Tipo de valor: Numérico.

## 3.4 Categoría D: UTILIDADES

3.4.1 Migración

Nombre del criterio: migrar esquemas

Descripción: permite usar esquemas de diseño de versiones anteriores

Tipo de valor: booleano.

3.4.2 Linq

Nombre de criterio: Estabilidad de Linq

Descripción: Permite a los desarrolladores escribir consultas contra el modelo conceptual mediante Visual Basic o Visual C#.

Tipo de valor:

3.4.3 Code First Mapping

Nombre de criterio: uso de code first

Descripción: permite definir un modelo usando clases de c# y visual basic.

Tipo de valor: booleano.

3.4.4 Recuperar conexión

Nombre de criterio: capacidad de recuperar la conexión

Descripción: capacidad para intentar automáticamente la reconexión al servidor por medio de todos los comandos disponibles.

Tipo de valor: booleano.

3.4.5 Soporte de operaciones asíncronas

Nombre del criterio: reparto del tiempo de ejecución

Descripción: permite bloquear y despertar hilos de ejecución de operaciones.

Tipo de valor: booleano.

# 4. Evaluación de los criterios por tecnología

## 4.1 Evaluación de los criterios para hibernate

Debe incluir al menos una tabla con la siguiente estructura.

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| CRITERIOS GENERALES |  |
| Criterio 1.1: Precio | Free(Gratis)  Comentario: es software libre |
| Criterio 1.2: Nombre |  |
| Criterio 1.3: Nombre |  |
|  |  |
| CRITERIOS REQUERIMIENTOS |  |
| Criterio 2.1: SGBD soportados | MSSQL  DB2  Ingres  Postgres  MySQL  Oracle  Sybase  Firebird  SQLLite |
|  |  |
| … |  |
|  |  |
| CRITERIOS RENDIMIENTO/EFICIENCIA |  |
| Criterio 3.1: TTL(Time to Load) | 2700ms |
| Criterio 3.2: SQ(Symple Query) | 49,5s  Comentario: se realizó una consulta simple (SQ) de una sola tabla que devuelve aproximadamente 30.000 entradas. Para lograr resultados realistas se realizó la consulta 100 veces. |
| Criterio 3.3: CQ(Complex Query) | 30,6s  Comentario: se utilizó una consulta compleja (CQ), que incluyó proyecciones (projections), uniones (joins), agrupación (grouping) y clasificación (sorting). |
| Criterio 3.4: EL(Eager Loading) | 50,2s  Comentario: tiempo empleado en realizar una eager loading |
| Criterio 3.5: CUD(cration,update & delete) | 3,3ms  Cometario: tiempo sacado a partir del objeto creado,al cual se le actualizan e insertan algunas porpiedades y son actualiazadas en la base de datos y finalmente son borradas de la base de datos, dicho procedimiento se repite 10.000 veces. |
| Criterio 3.6: CPU | 93.3%  Comentario: uso intensivo de la CPU |
| Criterio 3.7: RAM | 60.3MB  Comentario: uso de memoria moderada |
| Todas las operaciones anteriores, fueron realizadas en un ordenador con 3.0 GHz, 3GB de RAM, sobre Windows 7 Y SQL Server 2008. | |
| CRITERIOS DE UTILIDADES |  |
| Criterio 4.1: Migración | No  Comentario: sólo soporta la generación de un esquema inicial. Existe herramienta de código abierto de terceros (FluentMigrator), la cual podría utilizarse |
| Criterio 4.2: Linq | Sí  Comentario: para hibernate hay add-ons pero no tiene una perfecta integración, lo que produce errores. Los add-ons no son oficiales, son desarrollados por la comunidad. |
| Criterio 4.3: Code First Mapping | Sí  Comentario: sí soporta dicha función |
| Criterio 4.4: Recuperar conexión | Sí  Comentario: existen soluciones de terceros con NHibernate.SqlAzure |
| Criterio 4.5: Soporte de operaciones asíncronas | No  Comentario: Hibernate requeriría una importante refactorización de código y la arquitectura. No prevista a corto plazo. |

Y algunos comentarios aclaratorios sobre aquellos criterios cuyo valor indicado en la tabla no sea suficiente para entenderlo.

La tabla anterior es obligatoria y deben completarla los autores del trabajo, aunque se pueden incluir otros gráficos o tablas complementarias copiadas y pegadas desde diversas fuentes de información, siempre que debajo de cada uno se indique la fuente (al menos la URL).

## 4.2 Evaluación de los criterios de Entity Framework

|  |  |
| --- | --- |
| CRITERIOS | EVALUACIÓN |
| CRITERIOS GENERALES |  |
| Criterio 1.1: Precio | Free(Gratis)  Comentario: es software libre |
| Criterio 1.2: Nombre |  |
| Criterio 1.3: Nombre |  |
|  |  |
| CRITERIOS REQUERIMIENTOS |  |
| Criterio 2.1: SGBD soportados | MSSQL  MSSQL Compact  DB2  Informix  MySQL  PostgreSQL  Progress  Oracle  Ingres  Sybase  VistaDB  Sybase  SQLite  Firebird  Synergex  Virtuoso  Any Database with  ODBC or JDBC  driver |
|  |  |
| … |  |
|  |  |
| CRITERIOS RENDIMIENTO/EFICIENCIA |  |
| Criterio 3.1: TTL(Time to Load) | 230ms |
| Criterio 3.2: SQ(Symple Query) | 104s  Comentario: se realizó una consulta simple (SQ) de una sola tabla que devuelve aproximadamente 30.000 entradas. Para lograr resultados realistas se realizó la consulta 100 veces. |
| Criterio 3.3: CQ(Complex Query) | 44.6s  Comentario: se utilizó una consulta compleja (CQ), que incluyó proyecciones (projections), uniones (joins), agrupación (grouping) y clasificación (sorting). |
| Criterio 3.4: EL(Eager Loading) | 100s  Comentario: tiempo empleado en realizar una eager loading |
| Criterio 3.5: CUD(cration,update & delete) | 7.5ms  Cometario: tiempo sacado a partir del objeto creado, al cual se le actualizan e insertan algunas propiedades y son actualizadas en la base de datos y finalmente son borradas de la base de datos, dicho procedimiento se repite 10.000 veces. |
| Criterio 3.6: CPU | 98.6%  Comentario: uso intensivo de la CPU |
| Criterio 3.7: RAM | 61.9MB  Comentario: uso de memoria moderada |
| Todas las operaciones anteriores, fueron realizadas en un ordenador con 3.0 GHz, 3GB de RAM, sobre Windows 7 Y SQL Server 2008. | |
| CRITERIOS DE UTILIDADES |  |
| Criterio 4.1: Migración | Sí  Comentario: soporta dicha función después de su versión 6. |
| Criterio 4.2: Linq | Sí  Comentario: Integrado perfectamente. |
| Criterio 4.3: Code First Mapping | Sí  Comentario: soporta dicha función después de su versión 6. |
| Criterio 4.4: Recuperar conexión | Sí  Comentario: soporta dicha función desde su versión 6. La función de recuperar conexión automatiza el proceso de sentencias SQL y reintentan falladas. |
| Criterio 4.5: Soporte de operaciones asíncronas | Sí  Comentario: soporta dicha función desde su versión 6. |
|  |  |
|  | |

# 5. Comparación de las tecnologías

Debe incluir al menos una tabla resumen, en sección de página horizontal, cruzando los criterios y los valores de cada tecnología. Con una columna de comentarios sobre la comparación

Esta tabla anterior es obligatoria y deben completarla los autores del trabajo, aunque se pueden incluir otros gráficos o tablas complementarias copiadas y pegadas desde diversas fuentes de información, siempre que debajo de cada uno se indique la fuente (al menos la URL).

|  |  |  |  |
| --- | --- | --- | --- |
| CRITERIOS | HIBERNATE | ENTITY FRAMEWORK | COMENTARIOS |
| A.1 PRECIO | Free (Gratis) | Free (Gratis) | Ambas herramientas son gratuitas, es software libre. |
|  |  |  |  |
|  |  |  |  |
| B.1 SGBD | MSSQL  DB2  Ingres  Postgres  MySQL  Oracle  Sybase  Firebird  SQLLite | MSSQL  MSSQL Compact  DB2  Informix  MySQL  PostgreSQL  Progress  Oracle  Ingres  Sybase  VistaDB  Sybase  SQLite  Firebird  Synergex  Virtuoso  Any Database with  ODBC or JDBC  driver | Ambas soportan un gran número de sistemas gestores de bases de datos. Entity Framework soporta los mismo que Hibernate, además de una variedad mayor. |
| B.2 |  |  |  |
|  |  |  |  |
| C1 TTL (Time to load) | 2700ms | 230ms | Hibernate tiene un tiempo muy superior debido a la creación del objeto. Debido a que este proceso, se recomienda hacer esto solamente una vez en el tiempo de vida de una aplicación. |
| C2 SQ (Symple query) | 49,5s | 104s | Hibernate es el doble de rápido a la hora de la realización de consultas sencillas. |
| C3 CQ (Complex Query) | 30,6s | 44,6s | Hibernate es más rápido a la hora de realizar una consulta más compleja. |
| C4 EL (Eager Loading) | 50,2s | 100,0s | Hibernate es el doble de rápido. |
| C5 CUD () | 3,3ms | 7,5ms | Hibernate es 2,3 veces más rápido que Entity Framework |
| C6 CPU | 93,3% | 98,6% | Ambas realizan un uso intensivo de la CPU mientras están en ejecución, aun así es mayor en Entity Framework |
| C7 RAM | 60,3 MB | 61,9MB | Ambas tienen un uso de memoria moderado con muy poca diferencia entre ambas tecnologías. |
|  |  |  |  |
| D1 Migración | Sí\* | Sí | Entity Framework nos ofrece dicha función a partir de su versión 6, en cambio no nos lo ofrece aunque lo podríamos realizar a través de una herramienta externa de código abierto (Fluent migrator) |
| D2 Linq | Sí  Comentario: para hibernate hay add-ons pero no tiene una perfecta integración, lo que produce errores. Los add-ons no son oficiales, son desarrollados por la comunidad. | Sí  Comentario: Integrado perfectamente. |  |
| D3 Code first Mapping | Sí | Sí | Hibernate soporta dicha función y Entity Framework también a partir de su versión 6. |
| D4 Recuperar conexión | Sí\* | Sí | Hibernate si nos ofrece la función, pero es a través de terceros con NHibernate.SqlAzure, mientras tanto Entity Framework soporta dicha función desde su versión 6. La función de recuperar conexión automatiza el proceso de sentencias SQL y reintentan. |
| D5 Soporte de operaciones asíncronas | No | Sí | Entity Framework si soporta operaciones asíncronas, mientras Hibernate no puede y para incluir dicha función debería refactorización de código y la arquitectura, lo cual no tiene previsto realizar a corto plazo. |
|  |  |  |  |
|  |  |  |  |

# 6. Recomendaciones

Deben platearse posibles situaciones de uso, y recomendar justificadamente una u otra tecnología en función de la situación. Al menos 2 situaciones diferentes.

## 6.1 Situación 1

### 6.1.1 Descripción de la situación

*Una posible situación en el caso de comparar dos herramientas CASE, podría ser el caso de una empresa de desarrollo muy interesada en tecnologías open source, que programa sólo en Java, con equipos de desarrollo pequeños, que utiliza UML como notación, etc, etc*…

### 6.1.2 Recomendación de tecnología a utilizar

Debe indicarse la tecnología propuesta para esa situación.

Debe incluirse una tabla como la siguiente, mostrando las ventajas, respecto a los criterios, que ofrece cada tecnología en esa situación concreta.

Incluir sólo los criterios sobre los que se aprecien ventajas de una de las tecnologías frente a otra. No incluir criterios que no sean relevantes para la decisión (por ejemplo, el criterio “autor” seguramente no será relevante).

|  |  |  |
| --- | --- | --- |
| Criterios relevantes para la decisión | Ventajas tecnología 1 | Ventajas tecnología 2 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 6.2 Situación 2

### 6.2.1 Descripción de la situación

### 6.2.2 Recomendación de tecnología a utilizar

---------------------------

(Hay que cumplir la estructura básica indicada de secciones. Pero si se desea se pueden añadir otras secciones como anexos. Por ejemplo, alguna encuesta de opinión realizada sobre las tecnologías, etc.)