Revista Engenho, vol.11 –Dezembro de 2015

COMPARAÇÃO ENTRE OS FRAMEWORKS DE DESENVOLVIMENTO DE SOFTWARE '*ENTITY FRAMEWORK*' E '*NHIBERNATE*': ESTUDO DE CASO EM UM SISTEMA

Daniel José Angotti
Analista de Negócio, Repom S/A
djangotti@gmail.com

Carlos Eduardo Câmara
Professor do Centro Universitário Padre Anchieta
ccamara@anchieta.br

RESUMO

Neste artigo, através do desenvolvimento de software, realizamos a comparação entre *Entity Framework* e *NHibernate* – ambos *frameworks ORM* (Mapeamento Objeto-Relacional), que se comunicam com as bases de dados de aplicações desenvolvidas na linguagem .NET de maneira simples e praticamente invisível –, abordando as ferramentas disponibilizadas nesses *frameworks*, as quais auxiliam no processo de desenvolvimento de soluções para negócios. Foram realizados testes para comprovar qual dos *frameworks* utilizados propicia, ao negócio, um melhor desempenho com o sistema gerenciador de Banco de Dados utilizado, nesse caso, o *Microsoft SQL Server*. Com a aplicação desenvolvida, os resultados obtidos demonstraram que um dos frameworks se destaca, superando seu concorrente no desempenho de suas operações.

Palavras-Chave: *framework*, *Entity Framework*, *NHibernate*, *ORM*, Microsoft, *SQL*, *Server*.

ABSTRACT

Exposing a simple and objective way by creating a comparison between different applications Entity Framework and NHibernate, ORM frameworks (Object-Relational Mapping), which communicate with the databases of applications developed in .NET language simply and practically invisible. Addressing the tools available in these frameworks that assist in the development of solutions for business process. Conducted tests to prove which of the frameworks used to provide a better business performance management system with the database used in case the Microsoft SQL Server.

Keywords: framework, Entity Framework, NHibernate, ORM, Microsoft, SQL, Server.

1. INTRODUÇÃO

No cenário de desenvolvimento de softwares, uma das plataformas mais utilizadas é a .NET e, nesta, com intuito de aperfeiçoar e aumentar o desempenho das aplicações, novos *frameworks* foram desenvolvidos, contribuindo ainda mais com as novas aplicações disponibilizadas no mercado. Nesse contexto de novos *frameworks*, surgiram aqueles voltados para o mapeamento de objetos relacionais, os chamados *Object-Relational Mapping* (ORM), que realizam a abstração do Banco de Dados, facilitando o acesso da aplicação ao Banco de Dados.

Existem vários desses *frameworks* voltados à plataforma .NET (por exemplo, *Entity Cloud*, que é software livre), mas, neste trabalho, serão demonstrados dois dos mais conhecidos e utilizados no desenvolvimento de aplicações: o *Entity Framework*, criado pela própria Microsoft e disponibilizado para utilização em sua plataforma, e o *NHibernate*, *framework* de código livre, mantido pela instituição *NHForge* e derivado do *Hibernate*, utilizado em aplicações Java. Por serem os *frameworks* mais utilizados na plataforma .NET, esses são utilizados por um grande número de pessoas, para as quais fica a dúvida de qual seria o com melhor desempenho.

No desenvolvimento deste trabalho, serão utilizados autores como LERMAN (*Programming Entity Framework*) e PERKINS (*Working with NHibernate 3.0*), que tratam, cada um na sua especialidade, de boas práticas de arquitetura e desenvolvimento, utilizando esses *frameworks* ORM. Seus trabalhos auxiliam tanto no entendimento das ferramentas quanto em como desenvolvê-las, de maneira que o código tenha um ótimo desempenho e proporcione baixa manutenção.

Com as informações necessárias para o desenvolvimento, este trabalho apresenta, de forma simples, porém abrangente, como, o porquê e qual dos *frameworks* abordados tem um melhor desempenho na sua execução.

Inicialmente são apresentados os *frameworks*. Em seguida, é abordado o software utilizado para testar seu desempenho. Encerrando o presente trabalho, as considerações finais referenciam o software que se destacou e possui o melhor desempenho dentro de uma aplicação.

2. ORM – OBJECT–RELATIONAL MAPPING

Object-relational mapping ou mapeamento objeto-relacional é uma das técnicas utilizadas para realizar a abstração do Banco de Dados para a aplicação criada, dando a visão do Banco de Dados como objeto da aplicação ao invés de simples apresentação das tabelas do Banco de Dados. (DEVMEDIA, 2011).

O *ORM* é uma ponte, que realiza a ligação entre o mundo do "COMO" fazer, que são as classes de uma aplicação que dão suporte ao negócio, e o mundo do "O QUE" fazer, que retrata o mundo dos sistemas gerenciadores de Banco de Dados. A arquitetura de *ORM* está ilustrada na figura 1.

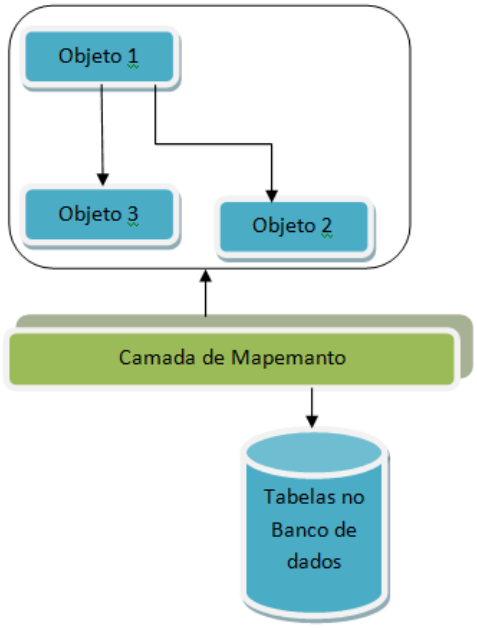

Figura 1 - Modelo ORM (DEVMEDIA, 2011).

As solicitações referentes à persistência, consulta, alteração e deleção dos dados do banco são realizadas através da aplicação, que é traduzida pela camada de mapeamento da arquitetura *ORM,* na qual é realizada a execução dos comandos do Gerenciador de Banco de Dados *SQL* diretamente na Base de Dados.

3. ENTITY FRAMEWORK

Framework desenvolvido pela Microsoft, integra, de modo nativo, ao *Visual Studio*, a partir da versão 3.5 do .NET. Baseado no modelo *ADO.NET*, o *Entity Framework* possui as mesmas características e funcionalidades daquele modelo, porém realiza o mapeamento dos objetos para, dessa forma, abstrair a Base de Dados e possibilitar um desenvolvimento mais simples ao desenvolvedor (MSDN, 2011).

Deste modo, o *Entity Framework* (*EF*) funciona conforme a arquitetura ilustrada na figura 2.

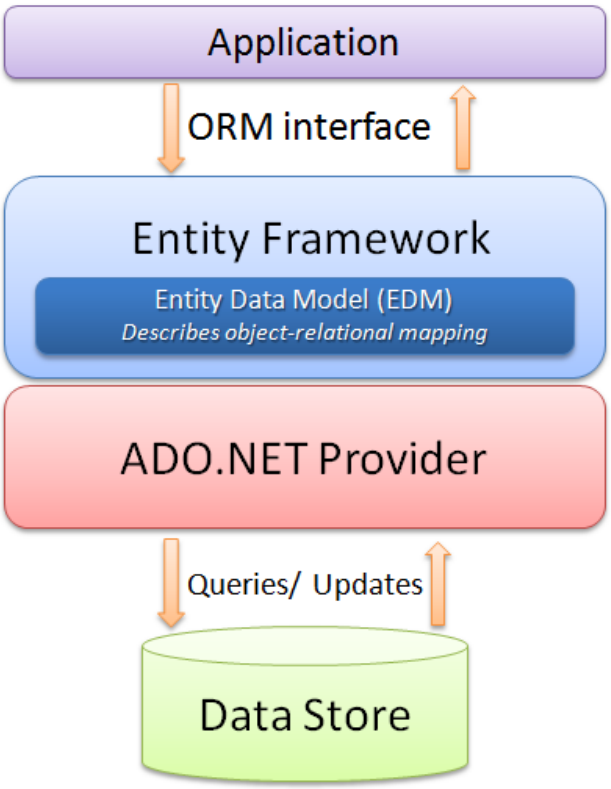

Figura 2 - Arquitetura Entity Framework (MSDN, 2014).

3.1.HISTÓRIA

A primeira versão do *Entity Framework* foi incorporada na versão 3.5 do .NET *Framework*, disponibilizada no *Visual Studio*, no ano de 2008, e provia, aos

desenvolvedores, o básico para realizar operações no padrão *ORM*, através da abstração da Base de Dados pela aplicação.

O *Entity Framework* conta com recursos para auxiliar no seu desenvolvimento e contribuir para o aumento de desempenho das aplicações que o utilizam. Nesse contexto, são utilizadas tecnologias como o ADO.NET, responsável por fornecer acesso a fontes de dados, como o Banco de Dados e os arquivos *XML,* através de *OLEDB* e de *ODBC*. Já o *Entity Data Model* (*EDM* ou, em português, Modelo de Dado e Entidade) utiliza, como base, o modelo de relacionamento entre entidades, descrito por Peter Chen em 1976. Mas, a partir desse, cria outros modelos e estende suas aplicações e utilizações tradicionais. O modelo entidade–relacionamento apresenta como ocorre o relacionamento de uma tabela com as demais, isto é, qual a forma de relacionamento – se é um relacionamento 1 para 1, 1 para N, N para N e assim por diante. (MSDN, 2014).

Outro recurso utilizado na melhoria de desempenho das aplicações é o *LINQ to Entities*, uma ferramenta que permite, aos desenvolvedores, escrever seus códigos de consulta ao EF, usando linguagens de desenvolvimento (VB.NET e C Sharp) no padrão *LINQ*. O *LINQ to Entities* realiza a conversão do código escrito, para realizar as operações através do framework *EF*. (MSDN, 2014). Existe, ainda, o *Entity SQL,* que é uma linguagem destinada à consulta de informações, independente do repositório utilizado. A consulta permite obter as informações como objetos ou em forma de tabelas. (MSDN, 2014).

4. NHIBERNATE

Como todo *framework ORM*, o NHibernate realiza a abstração do Banco de Dados para a realização de operações, como inserção, alteração, deleção e consulta de informações. Sua arquitetura está ilustrada na figura 3.

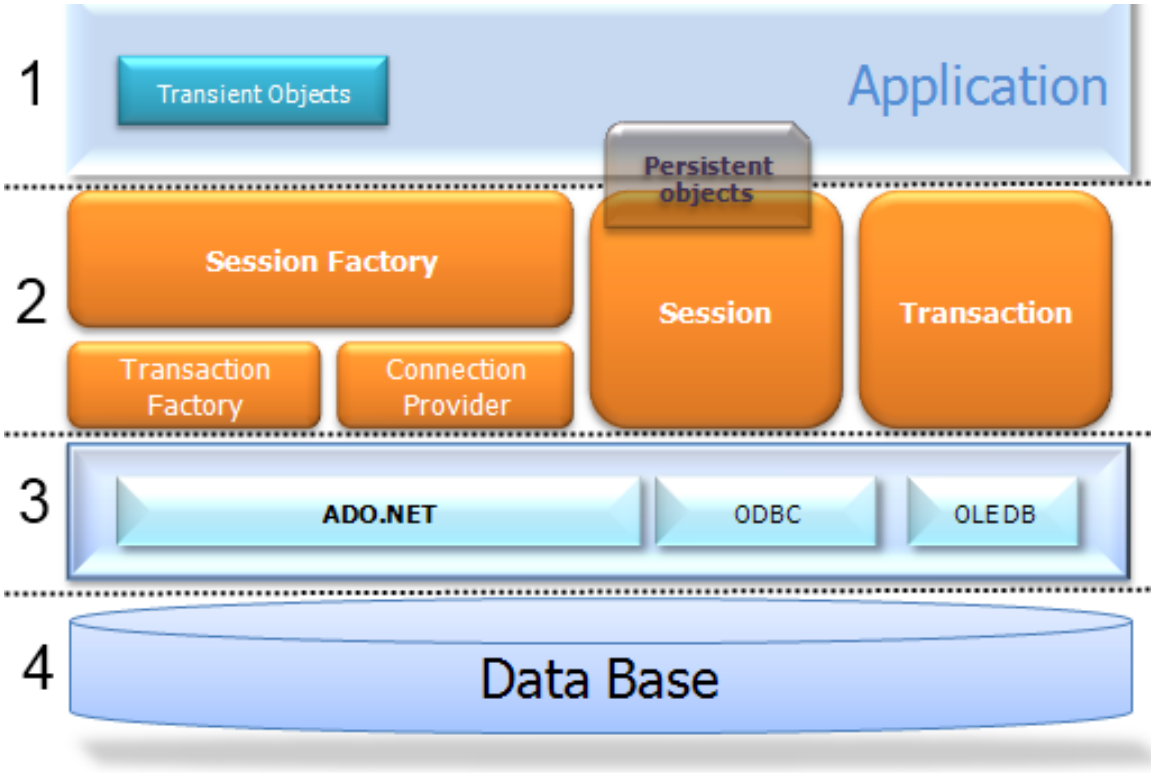

Figura 3 - Arquitetura NHibernate (DEVMEDIA, 2013).

Diferente das melhores práticas empregadas pelo *Enttity Framework* (EF), o *NHibernate* possibilita a criação de um modelo de dados a partir de seu código. É necessário criar as classes que serão utilizadas e, a partir dessas classes, criar ou não um Banco de Dados. Esse Banco de Dados terá características idênticas às dos modelos desenhados, como relacionamento, chave primária, tipos de dados, ações em cascata etc.(DEVMEDIA, 2013).

Enquanto o EF pode obter a Base de Dados para geração de suas classes, o *NHibernate* gera as classes para transformá-las em um Banco de Dados.

Além das características para abstração das informações, a maneira como são tratadas as classes em cada um dos cenários é diferente. Enquanto o *NHibernate* necessita de classes específicas para determinar as ações que serão realizadas (inclusão, consulta, alteração e deleção), o EF realiza as operações com métodos definidos para cada classe, para cada entidade em seu modelo lógico. (MSDN, 2014).

4.1. HISTÓRIA

Derivado do framework *Hibernate*, utilizado em Java, apresentamos o *NHibernate*, que é o *ORM* criado para atender à plataforma de desenvolvimento .NET.

Um dos recursos utilizados pelo *NHibernate* para auxiliá-lo é o *HQL,* que é uma linguagem idêntica ao *SQL* para realização de consultas através do *NHibernate*. Possibilita realizar consultas de maneira mais rápida e assertiva, utilizando as entidades mapeadas.

Segundo Aaron Cure e o Dr. Gabriel Schenker o *HQL* que é orientado a objetos, é uma forma valiosa de consultar dados e ainda é a API mais completa de todas. (CURE, SCHENKER, 2011).

Outra tecnologia utilizada pelo *NHibernate* é o *Lazy Load* ou inicialização tardia, é um recurso que permite que um determinado objeto seja carregado somente quando for utilizado, ou seja, realiza a consulta somente nos momentos necessários.

5. ESTABELECIMENTO DO PROBLEMA

Com base nas diferenças entre cada *framework* apresentado, foi realizada uma comparação para identificar qual deles consegue realizar ações em lote, com o melhor desempenho possível. Para isso, foram criadas aplicações idênticas, utilizando *frameworks* diferentes, que iriam realizar as ações de incluir, alterar, consultar e deletar.

O desenvolvimento das aplicações do sistema utiliza, como base, os recursos apresentados e discutidos anteriormente. Dessa forma, a aplicação 1 (que utiliza *Entity Framework*) contou com recursos, como *EDM*, *LINQ TO Entities* e *ADO*.NET *Provider*. Já a aplicação 2 (que utiliza *NHibernate*) contou com os recursos *Lazy Load* e *HQL*. Ambas as aplicações se conectam à mesma Base de Dados, propiciando, dessa forma, uma base de informações única, da qual é possível obter todo o tempo gasto em cada aplicação, para realizar as suas operações na Base de Dados.

Para obtenção do tempo de desenvolvimento em cada aplicação, foram incluídos cronômetros. Esses cronômetros são iniciados a partir do momento em que são chamados, ou seja, no momento em que ocorre a solicitação da aplicação para realização da operação.

O cronômetro é parado somente quando ocorre o "*commit*" ou, em termos mais conhecidos, a efetivação da transação, que caracteriza que a operação foi realizada com sucesso na Base de Dados.

6. ESPECIFICAÇÃO DO SISTEMA

O desenvolvimento dessas soluções utilizam a mesma base, ou seja, linguagem C# para *Windows Forms*, framework .NET 4.5 e como interface de desenvolvimento, foi utilizado o *Visual Studio 2013*.

Na aplicação 1 o *framework* Entity Framework 5 realiza o acesso a dados. Já na aplicação 2 do *framework NHibernate 4* acessa os dados para a aplicação.

O sistema gerenciador de Base de Dados utilizado foi o *Microsoft SQL Server* 2012.

6.1. MODELO DE DADOS

O Modelo de Dados, apresentado na figura 4, foi gerado a partir das tabelas criadas no Banco de Dados, e é consumido por ambas as aplicações.

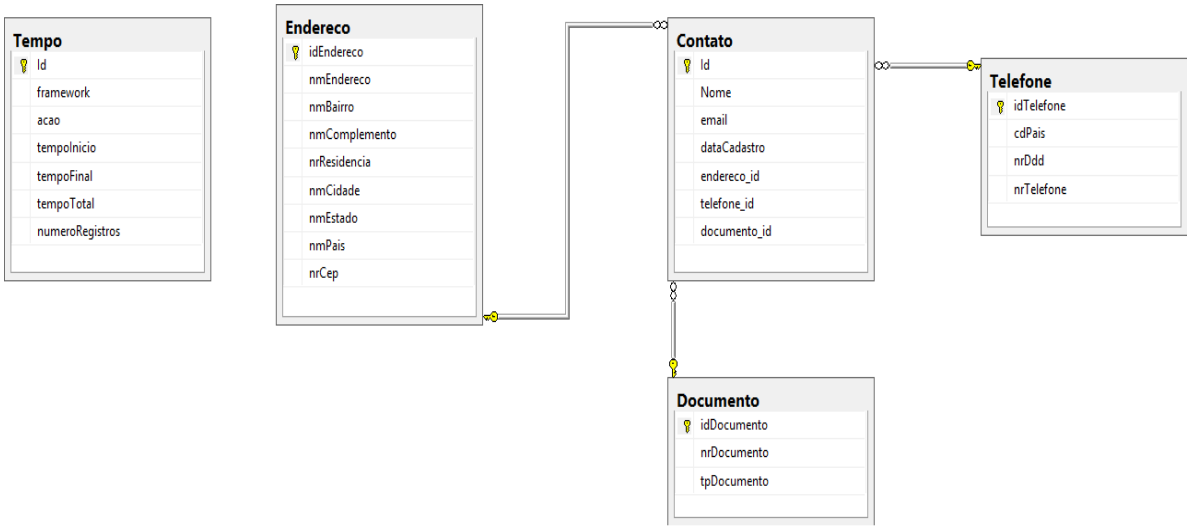

Figura 4 – Diagrama de Dados.

6.2. DIAGRAMA DE CLASSES

No *Diagrama de Classes* das aplicações desenvolvidas, foi utilizada a mesma base de classes para ambas as aplicações – o Diagrama de Classe da aplicação desenvolvida no *Entity Framework,* conforme mostra a figura 5:

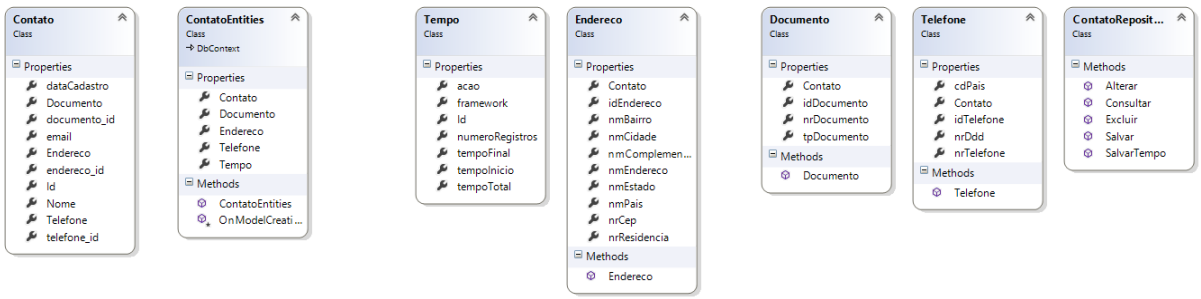

Figura 5 – Diagrama de Classe aplicação Entity Framework.

A figura 6 refere-se ao *Diagrama de Classes* da aplicação *NHibernate*, mantendo o mesmo padrão utilizado para a aplicação *Entity Framework*. As classes criadas para as diferentes aplicações possuem, praticamente, a mesma estrutura. Dessa forma, suas propriedades e seus metódos são, basicamente, os mesmos.

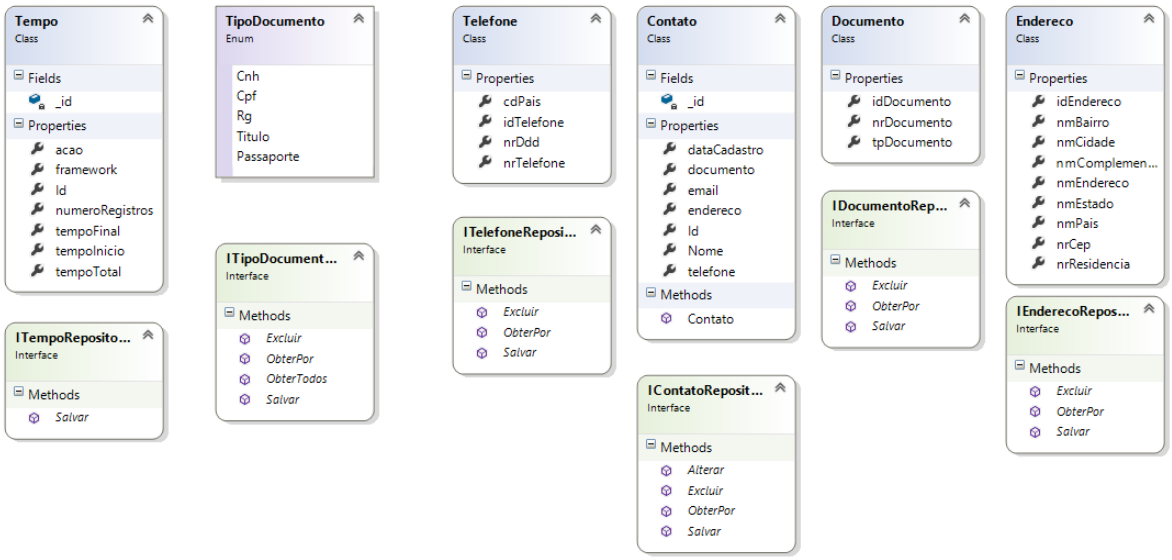

Figura 6 – Diagrama de Classe aplicação NHibernate.

6.3. PROTÓTIPOS DE TELA

Os protótipos de tela demonstram as telas em *Windows Forms,* utilizadas para realizar as ações em lote para as aplicações desenvolvidas. Idênticas, as telas auxiliam o usuário a visualizar o tempo realizado pela operação selecionada. Na figura 7, são demonstradas as quatro telas desenvolvidas para a aplicação com *Entity Framework*.

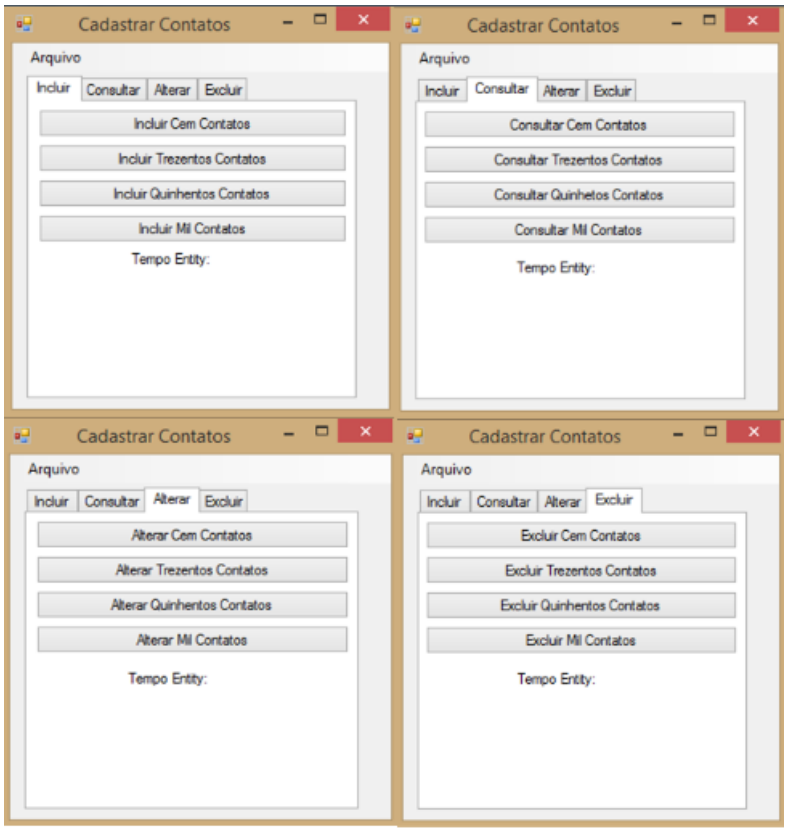

Figura 7 – Telas da Aplicação baseada em Entity Framework.

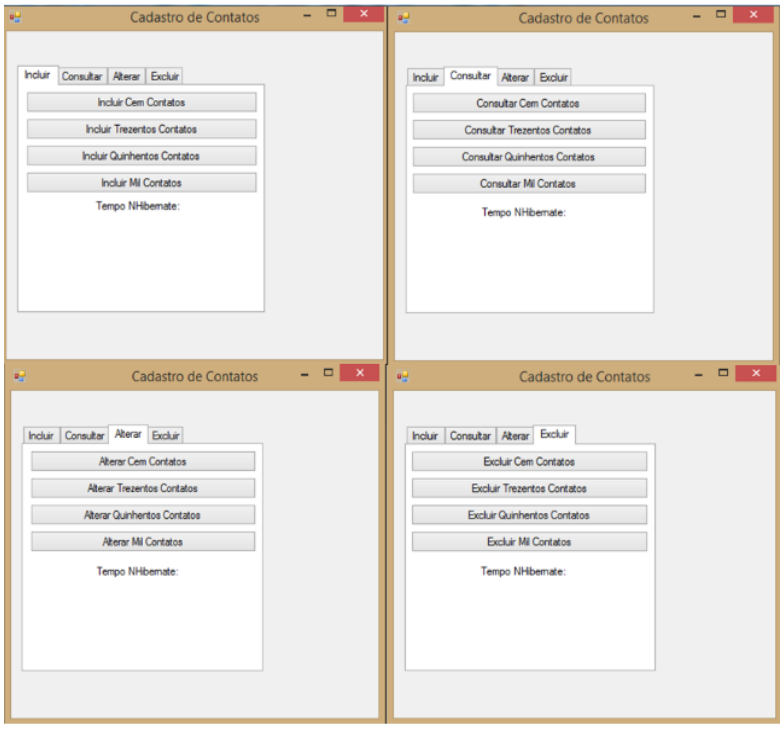

Figura 8 – Telas da Aplicação baseada em NHibernate.

Na figura 8, são mostradas as telas para a aplicação *NHibernate*. Como já afirmado, as telas possuem o mesmo padrão.

7. TESTES DO SISTEMA

Utilizando a metodologia de *Caixa-Cinza,* foram executados os testes de toda a aplicação. Os testes, conhecidos como *Caixa-Cinza,* são realizados de tal forma, que é possível analisar o código da aplicação, como pode ser observado no teste de *Caixa-Branca* e também nos testes de *Caixa-Preta,* nos quais são validadas as regras de negócio e a interface da aplicação. (DEVMEDIA, 2014).

Outro fator importante, ~~e~~ que influencia os testes realizados, é a máquina utilizada para a execução dos testes, a qual possui a seguinte configuração:

- Processador Intel, i7-4770 de 3.4GHz, com 4 núcleos;
- 8 GB de memória RAM
- Sistema Operacional Windows 8

7.1. CENÁRIOS DE TESTE

Para cada *software* desenvolvido, são utilizados os mesmos cenários de testes, proporcionando, dessa forma, a mesma visão para ambos os aplicativos.

➢ Inclusão de informações: são realizadas inclusões no sistema, em lotes. São esses lotes, de cem, trezentos, quinhentos e até mil “contatos”, que serão incluídos na Base de Dados.

➢ Consulta de informações: as consultas ao sistema são feitas em lotes. São esses lotes, que possuem cem, trezentos, quinhentos e até mil “contatos”, que serão consultados na Base de Dados.

- Alteração de informações: são realizadas alterações no sistema em lotes. São esses lotes, de cem, trezentos, quinhentos e mil “contatos”, que serão alterados na Base de Dados.

- Exclusão de informações: são realizadas exclusões no sistema em lotes. São esses lotes, de cem, trezentos, quinhentos e mil “contatos”, que serão excluídos na Base de Dados.

7.2. RESULTADOS DE TESTE

Abaixo, seguem os resultados dos testes realizados com cada uma das operações – inclusão, consulta, alteração e exclusão – das aplicações 1 e 2.

Inclusão de informações para cada aplicação:

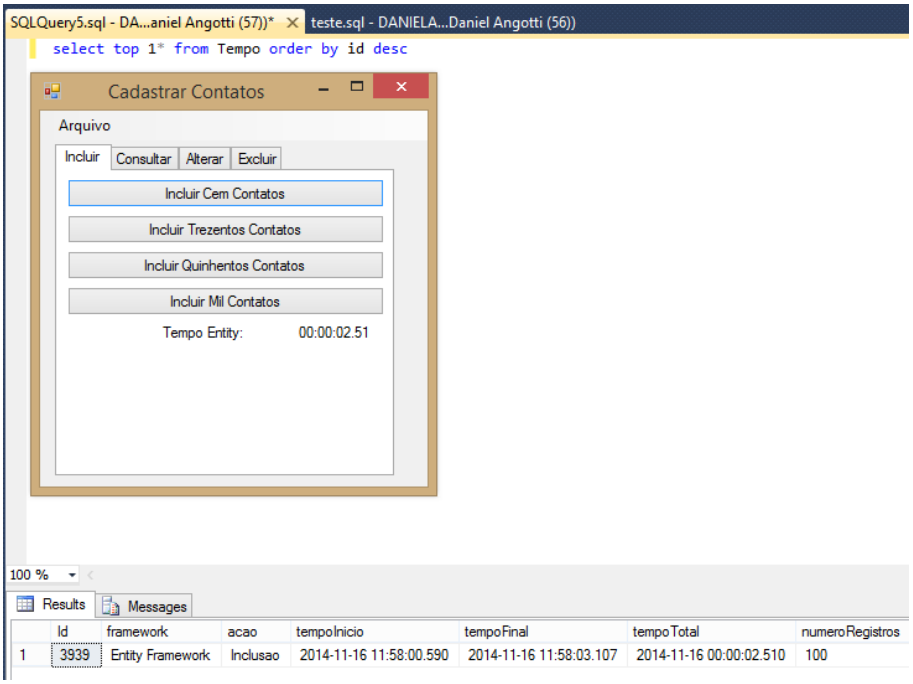

Figura 9 – Inclusão de informação aplicação 1.

A figura 9 mostra que a inclusão da Aplicação 1 foi realizada, e o seu tempo foi mostrado ao usuário e também armazenado na Base de Dados.

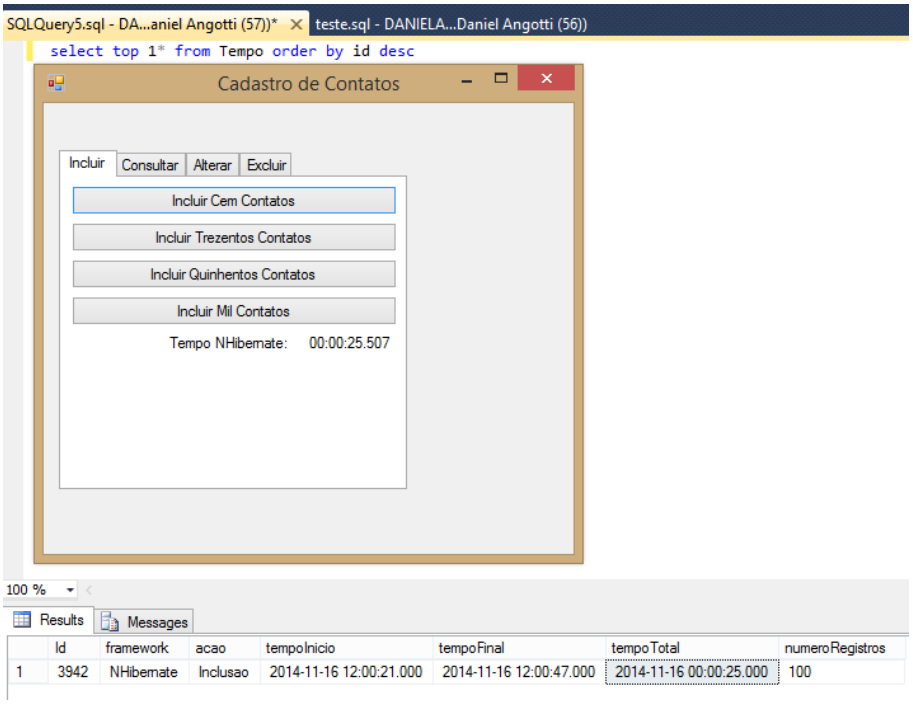

Figura 10 – Inclusão de informação aplicação 2.

Na figura 10, a evidência mostra que a inclusão da Aplicação 2 também foi realizada, o seu tempo foi mostrado ao usuário e também armazenado na Base de Dados.

Alteração de informações para cada aplicação foi apresentada nas figuras 11 e 12, nas quais é possível visualizar que as informações foram alteradas na Base de Dados e o seu tempo foi armazenado.

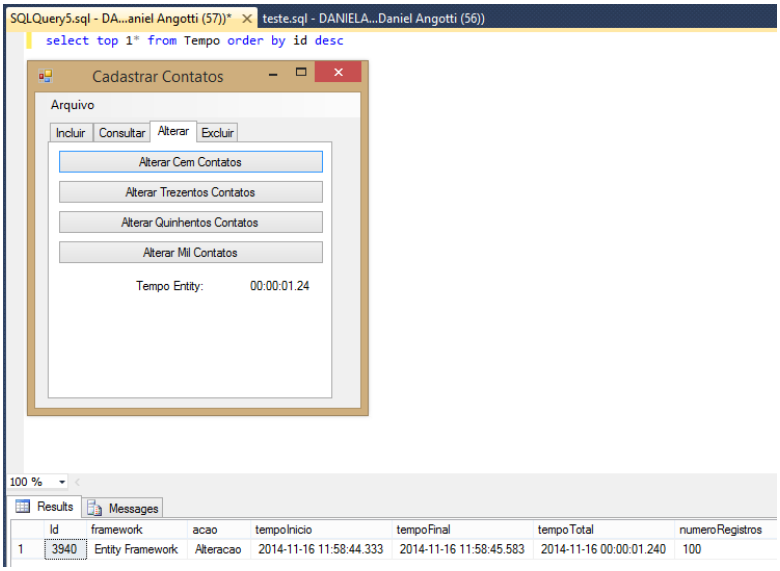

Figura 11 – Alteração de informação aplicação 1.

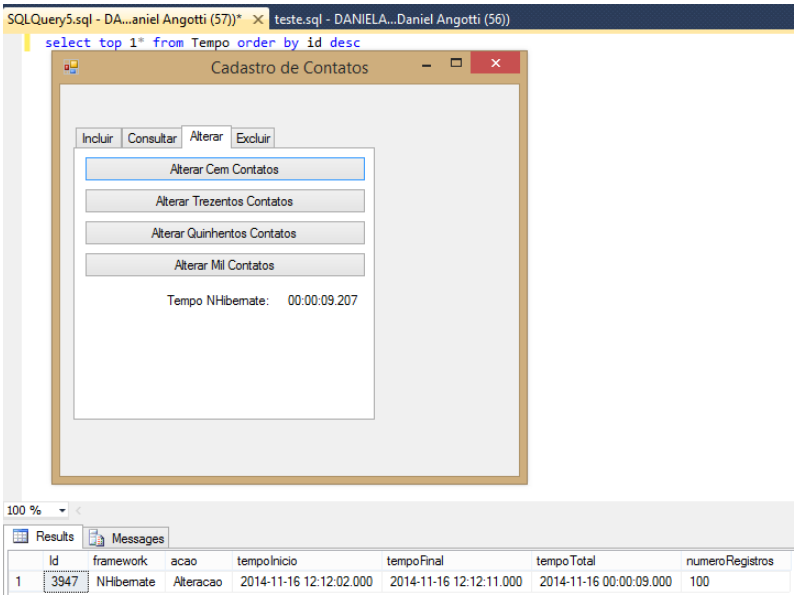

Figura 12 – Alteração de informação aplicação 2.

Nas figuras 13 e 14, é possível ver a exclusão de informações para cada aplicação e o armazenamento de seu tempo.

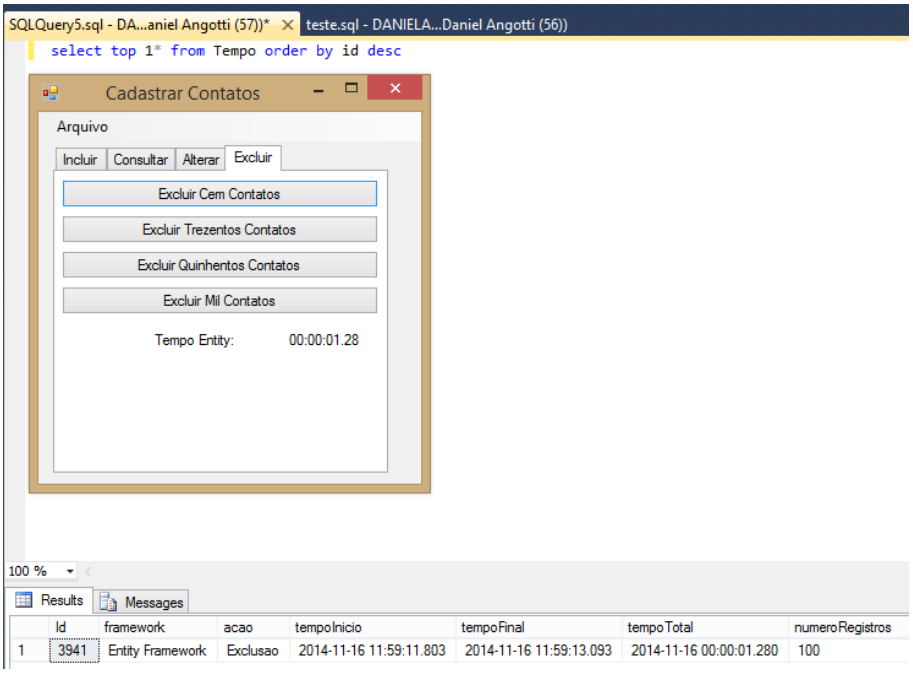

Figura 13 – Exclusão de informação aplicação 1.

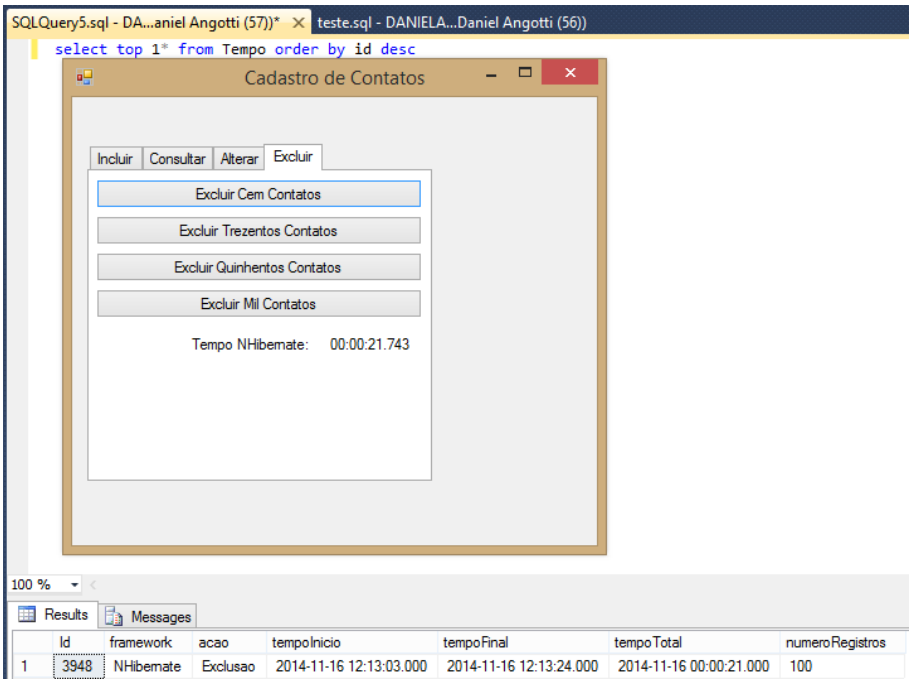

Figura 14 – Exclusão de informação aplicação 2.

Os testes realizados mostram que os cenários desenvolvidos foram atendidos. Conforme os testes realizados, constatou-se que a aplicação com o *Entity Framework*

superou em performance e em tempo de execução. Conforme visto na figura 15, a aplicação que utiliza o *NHibernate*, possui um desempenho superior. Os testes realizados conseguiram provar a proposta do trabalho, porém, para as operações de consulta a obtenção dos resultados, foi dificultada, já que a aplicação *NHibernate* não conseguiu capturar os milissegundos gastos para realizar tal ação.

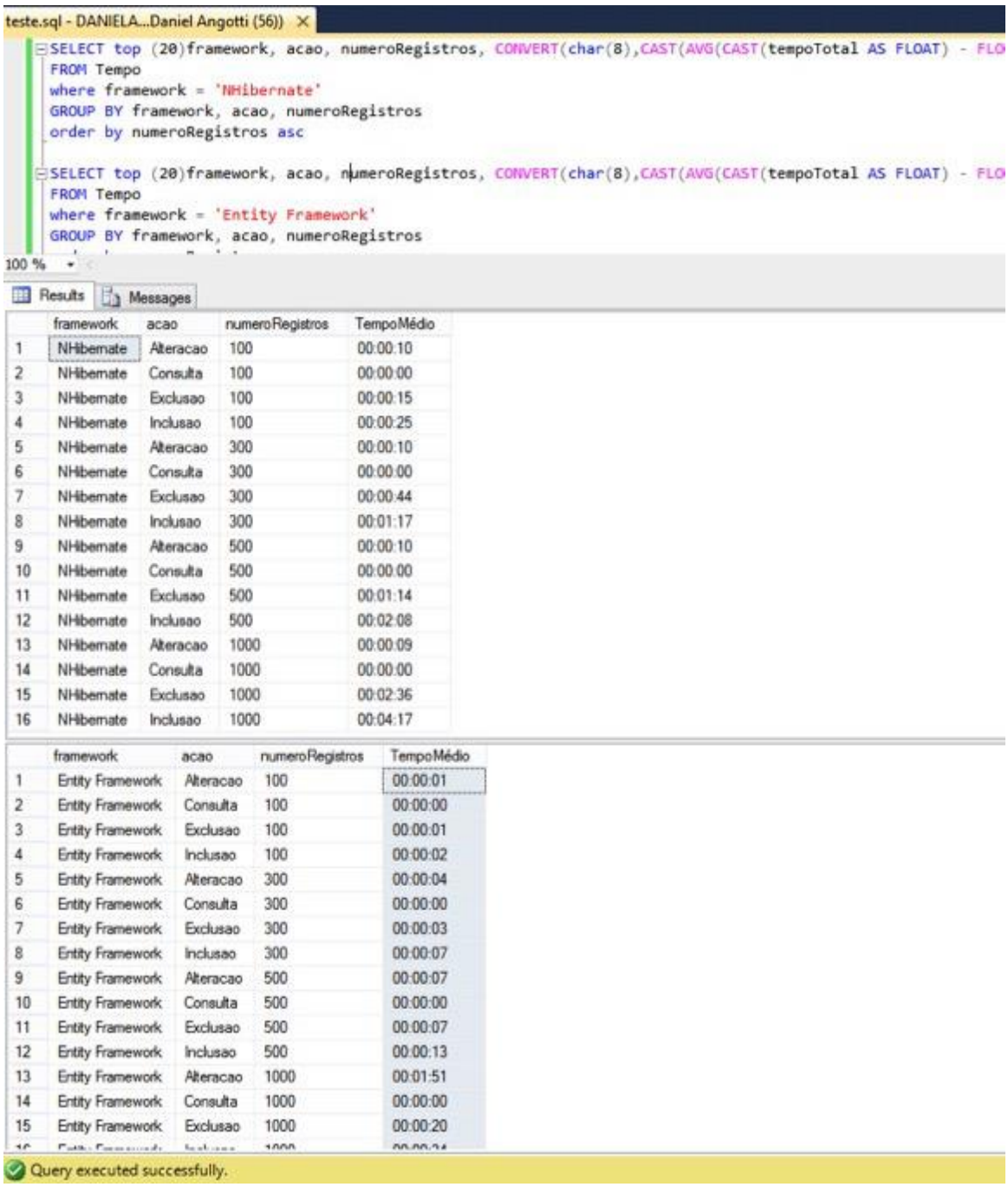

Figura 15 – Tempo médio por aplicação e operação (ação).

É possível visualizar, através de um comando no Banco de Dados (comando de seleção de informações), os dados inseridos na Base de Dados do projeto e identificar o desempenho de cada ação.

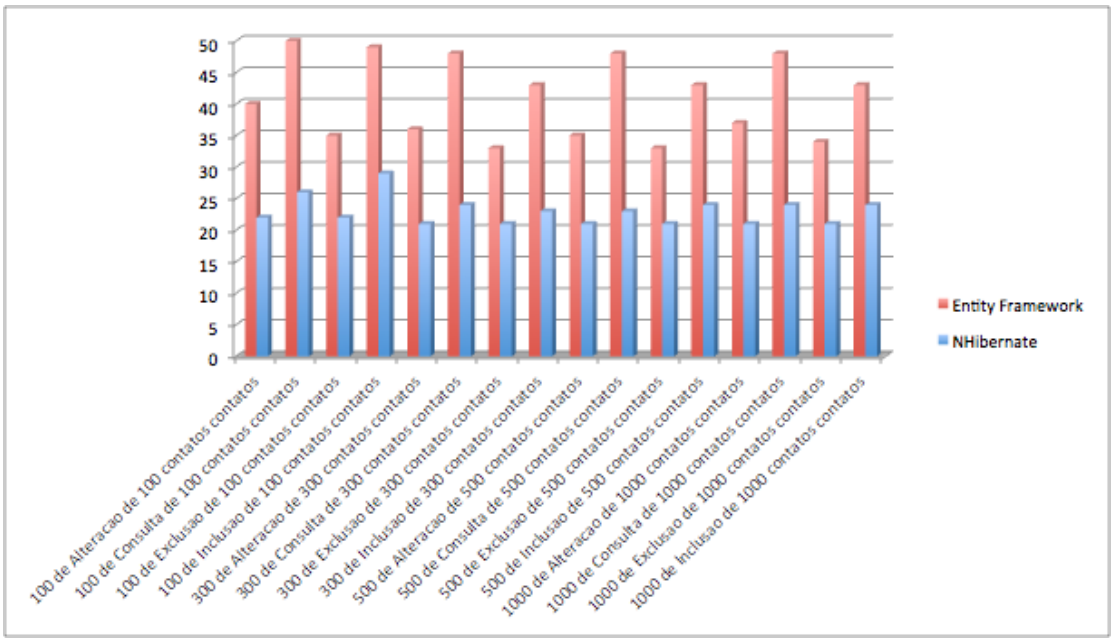

Figura 16 – Gráfico com o número de testes realizados.

Gráfico para melhor visualização e disposição das informações.

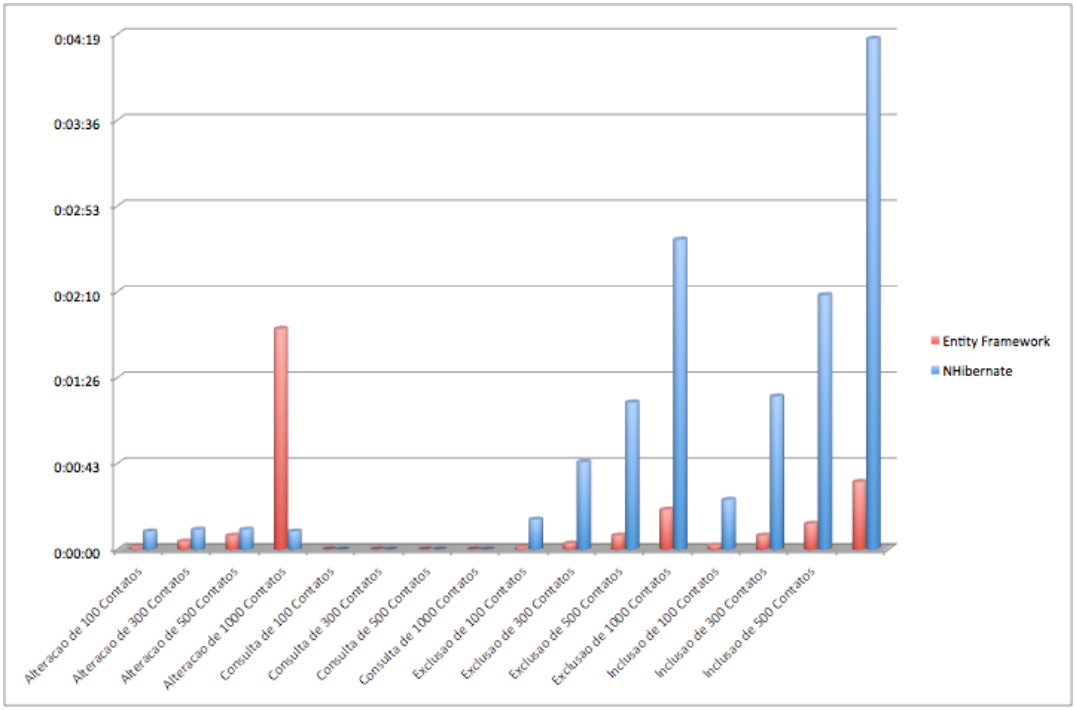

Figura 17 – Gráfico com o tempo médio por aplicação e operação (ação).

As aplicações desenvolvidas não permitiram a execução de maiores lotes de informações. Quando executadas para realizar ações com um número maior de 5 mil linhas em cada tabela, ocorria um erro de *Time Out*. Esse erro ocorre quando não é possível realizar transações na Base de Dados, por conta de um esgotamento de tempo de conexão com o banco.

8. CONSIDERAÇÕES FINAIS

Após a realização dos testes, foi identificado que o *framework* que atinge o melhor desempenho para realização das operações de inclusão, alteração, consulta e deleção é o *Entity Framework*, que conseguiu alcançar um desempenho, para algumas operações, até 10 vezes mais rápido.

Um dos pontos analisados – e que pode, ocasionalmente, ter causado a vantagem do *Entity Framework* – foi que o código para uma ação na Base de Dados, utilizando o framework *NHibernate*, constrói toda a estrutura necessária para realizar uma ação na Base de Dados, gerando linhas e mais linhas de código para realizar a mesma operação.

Outro ponto analisado foi a integração nativa entre o *Visual Studio* e o *Entity Framework*, facilitando o desenvolvimento e a comunicação com a Base de Dados.

O resultado foi alcançado devido às tecnologias empregadas e discutidas neste trabalho, auxiliado pela abstração do Banco de Dados, e por integrar, de forma consistente, a base de ferramentas da Microsoft.

O trabalho desenvolvido alcançou seu objetivo, apresentando, de maneira simples e objetiva, a aplicação de ferramentas *ORM*, através dos recursos das ferramentas *Entity Framework* e *NHibernate,* e a comparação entre essas duas principais ferramentas da plataforma de desenvolvimento .NET.

REFERÊNCIAS BIBLIOGRÁFICAS

CURE, A.; SCHENKER, G. N.; **NHibernate 3.0 Beginner's Guide**. Birmingham: Packt, 2011. 368p.

DENTLER, J.;. **NHibernate 3.0 Cookbook**. Birmingham: Packt, 2010. 315p.

DEVMEDIA. **NHibernate: Arquitetura, fundamentos e recursos.** Disponível em <http://www.devmedia.com.br/nhibernate-arquitetura-fundamentos-e-recursos-revista-net-magazine-102/26942>, recuperado em 18/11/14.

NHFORGE, NHibernate. **NHibernate Reference Documentation.** Disponível em < http://nhforge.org/doc/nh/en/index.html> recuperado em 29/03/14, atualizado para http://stc.sbu.ac.ir/AdminTools/Docs/Files/nhibernate_reference.pdf em 2015.

LERMAN, J.;. **Programming Entity Framework**. . Sebastopol: O'REILLY, 2010. 873p.

MACORATTI, J. C.; **Entity Framework - Sem Firulas : Histórico de versões.** Disponível em <http://www.macoratti.net/14/03/ef_hver1.htm>, recuperado em 18/11/14.

MICROSOFT. **Entity Framework Documentation.** Disponível em <http://msdn.microsoft.com/pt-br/data/aa937709 >, recuperado em 29/03/14.

PERKINS, B.; **Working with NHibernate 3.0**. Indianapolis: Wiley Publishing, 2011. 221p.

SOURCEFORGE. **NHibernate.** Disponível em < http://sourceforge.net/projects/nhibernate/> recuperado em 21/11/14.