ANÁLISE COMPARATIVA ENTRE ENTITY FRAMEWORK E NHIBERNATE

Web DevelopmentUsingNHibernate

Celso de Ávila Ramos[1]

Caio de Oliveira AmoêdoPinelli; Eric da Silva Cardoso; Tamara Regina Moreira[2]

[1]Docente no Curso da Ciência da Computação da Universidade José do Rosário Vellano - UNIFENAS, câmpus Alfenas, MG, Brasil. <celso.ramos@unifenas.br>.

[2]Discentes Graduados no Curso de Ciência da Computação da Universidade José do Rosário Vellano - Câmpus Alfenas, MG, Brasil. <caiopinelli89@hotmail.com>,<eric_674@msn.com>,<tamara.rm28@gmail.com>.

Resumo:

O trabalho visou avaliar a agilidade, complexidade, confiabilidade e outros tantos atributos das novas tecnologias de acesso a dados *Entity Framework* e NHibernate. Consta de um estudo descritivo e de um software para demonstrar satisfatoriamente a comparação. O foco principal foi na complexidade de implementação do software e na agilidade no acesso a dados, que hoje em dia no mercado de trabalho é primordial. O software feito é de um controle de condomínio que faz cadastros de clientes hospedados, como também o cadastro dos quartos ocupados por cada cliente. Ao final do projeto uma inserção de dados foi feita ao banco de dados, podendo assim concretizar a pesquisa, a fim de conseguir demonstrar qual tecnologia sobressai em termos de agilidade no acesso aos dados.

Palavras-Chave: *Entity Framework*, NHibernate, Tecnologia.

Abstract:

Thestudy aims to evaluatethe agility, complexity, reliability, and many other attributesof new technologies fordata accessEntity FrameworkandNHibernate. It consists of adescriptive study ofa software tosatisfactorily demonstratethe comparison. The main focuswason the complexityofthe software implementationandflexibility indata access, which in today'slabor marketis paramount.Thesoftwareismadeof acondominium thatmakescontrolregistersof hostedcustomers, as well asthe registration ofeach clientoccupied byrooms.At the endof the projectan insertion ofdata was madeto the database, thus being able tofinishthe surveyin order tobe able to demonstratewhat technologyexcelsin agilityin accessing data.

Keywords: Entity Framework, NHibernate, Technology.

Introdução:

Com o aumento do desenvolvimento Web, surgem as dificuldades de programação de códigos grandes e complexos para acessar banco de dados. Para sanar essas dificuldades, novas tecnologias de acesso a dados são criadas e uma delas é o uso do *NHibernate*.

NHibernate é um Framework Open Source baseado em.NET para persistir os objetos para bases de dados relacionais, baseado em uma ferramenta de persistência de dados do Java(Hibernate), que possibilita o mapeamento de objetos-relacionais(ORM), podendo ser usado em variados bancos de dados. Ao contrário das versões da Microsoft de acesso a dados DAO, RDO, ADO, ADO.NET, *Entity Framework* o *NHibernate*, juntamente com o Visual Studio (o compilador C# e de outras linguagens), o *NHibernate* utiliza arquivos XML para efetuar o mapeamento entre as propriedades da classe e a tabela desejada no banco de dados, pois o *NHibernate* gera o código SQL necessário. Desenvolvendo um simples sistema web baseado em um controle de condomínio, deseja-se mostrar quais as diferenças relacionadas ao *NHibernate* e o *Entity Framework*, as vantagens e desvantagens do uso da ferramenta *NHibernate*.

O ADO.NET *Entity Framework* é uma plataforma de acesso a dados desenvolvida pela Microsoft e primeiramente incorporado no *Service Pack* 1 do Visual Studio 2008 e .NET Framework 3.5. A Microsoft desde então, tem investido bastante neste poderoso Framework, e atualizações constantes são feitas.(BALTIERI, 2012)

O principal benefício do ADO.NET *Entity Framework* é sua alta produtividade, visando que não é preciso se preocupar com o modelo de dados, tudo será gerado a partir de ferramentas integradas no Visual Studio. Os modelos gerados possuem entidades que representam as tabelas do banco de dados, sendo assim, trabalhamos de forma totalmente orientada à objetos.

De acordo com Durães (2010), usando a linguagem *LINQ ToEntity Framework*, poderá ser feito mapeamentos para diversas bases de dados (*SQLServer*, *Oracle*, *DB2,MySql, PostgreSQL, SQLite, VistaDB, Informix, Sybase, etc*.) assim como para outras fontes como *XML* e serviços.

Segundo Pandolfo (2010), com a utilizaçãodo *Entity Framework*, a preocupação não é mais com o banco de dados, a atenção se fixa na camada de negócios e nas entidades que refletem o modelo de dados. Os resultados são retornados como objetos, e os programadores não precisam perder muito tempo fazendo códigos referentes aos bancos de dados.

O EDM descreve o banco de dados, o modelo de classes e seu mapeamento (que fornece os metadados necessários para que o *Entity Framework* possa resolver os problemas de impedância). Esse mapeamento expresso no EDM possibilita que, por exemplo, não seja necessária a utilização de uma mesma convenção de nomes para tabelas e classes, desamarrando completamente aplicação e fonte de dados. É composto por três artefatos feitos em *XML* (*.CSDL, *.MSL e *.SSDL). Em tempo de execução, eles são analisados e seus dados armazenados em classes, que podem ser consultadas no processo de materialização.

O Visual Studio tem um assistente que gera automaticamente as informações de mapeamento do banco de dados e classes de negócio, que podem ser modificadas através de um designer visual. Para facilitar a integração com o visual Studio, o time de desenvolvimento do *Entity Framework* criou a extensão EDMX, que é composta pelos três arquivos XML do EDM. Para que a estrutura de acesso a dados funcione corretamente e

acesse os dados é preciso usar um arquivo para guardar a string de conexão, o *App.Config*, que é o arquivo onde fica armazenado os dados da localidade do banco de dados e os arquivos de mapeamento gerados pelo nosso *EDM.*

A string de conexão é automaticamente anexada no arquivo ***APP.CONFIG***. Se existir mais de um modelo EDM, será anexado automaticamente à este arquivo.

Segundo Coimbra (2010), o Object Services é a camada responsável pelo gerenciamento das entidades do *Entity Framework*, expondo uma API completa para a geração de consultas que materializam objetos, além de capacidades CRUD (Inserir, Remover, Alterar e Deletar).

LINQ *to Entities* é um dialeto do LINQ (*Language Integrated Query*), e permite que o desenvolvedor escreva consultas baseadas no modelo de dados a partir da mesma linguagem utilizada para construir a lógica de negócio. Um benefício muito importante conseguido com o seu uso, é a não necessidade de se lembrar do nome exato das tabelas e atributos do banco de dados, deixando a cargo do *IntelliSense*(mostra automaticamente os nomes das classes, tabelas e variáveis existentes) do Visual Studio a tarefa de descobri-las, e ainda tendo a ajuda do compilador para checar possíveis erros. Mesmo que o LINQ *to Entities* seja operado em objetos, ele ainda precisa ser traduzido para SQL, que finalmente é lançada no banco de dados.

Muitos dos métodos e sobrecargas do LINQ não podem ser representados em SQL. Apesar disso, são sintaticamente válidos, e uma chamada a um deles não irá causar erro em tempo de compilação, mas sim uma *NotSupportedException* em tempo de execução. (Em: <http://blog.guaratech.com.br/2011/10/*Entity*-framework.html>. Acesso em: 13 outubro 2012).Segundo Pereira (2010), a expressão lambda pode ser definida como uma função onde ela pode conter algumas expressões ou instruções. Todas as expressões lambda usam o operador lambda (= >), que é lido como "vai para". O lado esquerdo do operador lambda especifica os parâmetros de entrada (se houver) e o direito contém a expressão ou o bloco de instruções. A expressão lambda x => x * x é lido " x recebe x vezes x. ". As expressões Lambdas são usadas em métodos baseados em consultas LINQ como argumentos para métodos de operador de consulta padrão como Where.

Estas expressões podem ser atribuídas a uma consulta da seguinte maneira:

```
var consulta = consulta(c => c.id == 1).toList();
```

Esta consulta realizada retorna uma lista, a expressão lambda faz a consulta sendo o “c” o objeto da tabela referente à consulta deseja e o id seria o campo da tabela.

Segundo Aéce (2004), *NHibernate* é uma biblioteca (Framework) baseada em .NET para persistir os objetos para bases de dados relacionais. Baseado em uma ferramenta de persistência de dados do Java, chamado Hibernate, o *NHibernate* tem a finalidade de persistir os objetos .NET em uma base de dados relacional subjacente. Isso facilita muito ao invés de escrever códigos SQL dentro e fora da base de dados.

Segundo Thomaz (2011), O *NHibernate*utiliza ORM’spara mapeamento das tabelas. Com ele podemos criar uma estrutura de classes sem se preocupar com os famosos SELECT’s, INSERTS’s, UPDATE’s e DELETE’s, a própria ferramenta criará esses comandos em tempo de execução.

Segundo Macoratti (2011), Atualmente os principais banco de dados suportados pelo *NHibernate* são: Microsoft SQL Server 2008/2005/2000, Oracle, Microsoft Access, Firebird, PostgreSQL, DB2 UDB, MySQL, SQLite. O arquivo de configuração define as informações do provedor, dialeto, driver e string de conexão do banco de dados usado para persistência. Segundo Thomaz (2011), ORM é uma sigla em inglês que significa ObjectRelationalMapper (Mapeador Objeto Relacional). ORMs são ferramentas responsáveis por mapear as classes do modelo Orientado a Objetos para tabelas do modelo relacional. Através desse mapeamento, conseguimos dizer que determinada propriedade da classe será associada a um campo de uma tabela. Podemos reunir um conjunto de propriedades de várias classes e salvar na mesma tabela.Segundo Thomaz (2011), O *NHibernate* funciona como uma camada entre o modelo de classes e o modelo relacional. O *NHibernate*, entre outras coisas, cria os quatro comandos INSERT, SELECT, UPDATE e DELETE em tempo de execução, para isso, ele utiliza os recursos de Reflection (obtenção de informações em tempo de

execução) do .NET Framework e arquivos XML para mapeamento. Segundo Simas (2013), podemos observar que o *NHibernate* é acoplado diretamente entre O Banco de Dados a sua aplicação, porém, podemos observar também que os objetos persistentes - persistentobjects - são utilizados através da aplicação e, através de conexão com o *NHibernate*, nos dá a entender que são as peças mais importantes desta engrenagem. Podemos também observar que dentro do domínio do *NHibernate* vemos o mapeamento em arquivos XML - Seção XML Mappings e o arquivo de configuração do .NET, App.config ou Web.config. Fique certo de que o *NHibernate*, como uma Camada de Acesso aos Dados e, posteriormente de persistência, de sua aplicação, estão fortemente acoplados porque todas as outras camadas, caso existam, são totalmente dependentes destes objetos de persistência para E/S de dados. Seria muito interessante mostrar uma visão mais detalhada da arquitetura em tempo de execução do *NHibernate*, porém isso não é possível pelas formas diferentes de abordagens e configurações, destas separamos duas: As arquiteturas simples e detalhada.

Na sua arquitetura simples, faz com que a aplicação disponibilize suas próprias conexões utilizando o ADO.NET e gerencie suas próprias transações. Esta abordagem usa um pequeno conjunto de APIs do *NHibernate*.

Segundo Simas (2013), existem 4 regras principais para Classes POCO serem “reconhecidas” pelo *NHibernate*:

- Use propriedades para campos persistentes.
- Implemente um construtor padrão, ou seja, vazio.
- Crie uma propriedade Identificadora, ou Id.
- Utilize classes que não sejam sealed e use métodos virtuais.

Segundo Simas (2013), atualmente, o mapeamento no *NHibernate* pode ser executado utilizando um arquivo XML ou com um novo projeto onde são utilizados atributos para decorarem as classes, primeiramente, vamos ao mapeamento utilizando o XML.Segundo Simas (2013), o Mapeamento XML é a primeira etapa para a persistência com o *NHibernate* é onde tudo começa. Por motivos de compilação, o *NHibernate* solicita que os arquivos para mapeamento contenham também a extensão .hbm como forma de ser reconhecido pelo assembly do ORM. Estes arquivos precisam ter sua

propriedade Build Actionsetados como EmbeddedResource, posto que no momento da compilação do código, o arquivo seja reconhecido e convertido para um hierarquia de classes utilizando ponto, por exemplo: temos que mapear a Classe Pessoa para persistência, então criamos o arquivo pessoa.hbm.xml e nas suas propriedades, colocamos o atributo Build Action como EmbeddedResource.

Metodologia:

O presente estudo tem o propósito de demonstrar e explicar algumas das características do *NHibernate*, fazendo um comparativo com o *Entity Framework* com o objetivo de demostrar qual é o melhor método e qual é o mais rápido de se programar. O estudo realizará o desenvolvimento de uma aplicação utilizando *NHibernate* e *Entity Framework*.

Neste estudo serão analisados os seguintes itens: Agilidade de programação e velocidade de acesso a dados. Para obter uma resposta mais precisa deste estudo, foram feitas várias pesquisas em livros, trabalhos científicos, artigos e revista visando compreender os fundamentos e funcionalidades do *NHibernate*.

Foi desenvolvido um *software* com o propósito de demonstrar as funcionalidades do *NHibernate* usando o Visual Studio. Em algumas partes do desenvolvimento deste software, também será colocado partes do *Entity Framework* com o intuito de demonstrar a diferença entre as duas formas de acesso a dados, onde será feito o análise de qual tem o ganho em agilidade, performance e simplicidade. Com isso, espera-se comprovar que o *NHibernate* tem uma melhor eficiência comparado com o *Entity Framework.*

A linguagem de programação utilizada em conjunto com a plataforma ASP .NET foi o C#, que é uma linguagem de programação orientada a objetos criada pela Microsoft. O C# herda grande parte dos melhores recursos do C++ e Microsoft Visual Basic e pouco das inconsistências e anacronismos, resultando em uma linguagem mais limpa e lógica.

O C# desempenha um papel importante na arquitetura do Microsoft .NET Framework, sendo comparado, algumas vezes, à função que o C desempenhou no desenvolvimento do UNIX.

Neste presente estudo foi desenvolvido um *software* operacional via web para avaliar o desempenho do uso da ferramenta de acesso a dados *NHibernate* e compará-lo com a ferramenta *Entity Framework*.

Em visita em um condomínio fechado, notaram-se algumas dificuldades no registro de seus clientes, com um método muito arcaico para guardar os dados de seus clientes, sendo este, o uso de pastas e arquivos.

Foi proposto ao dono do condomínio a criação de um *software* para o registro dos clientes, podendo ser via web ou desktop. Neste caso foi optado via web, para facilitar e agilizar o uso do sistema.

O sistema deverá possuir os dados pessoais do cliente, telefones úteis, alas do condomínio, quartos de cada ala do condomínio e o cadastro do cliente a um quarto do condomínio.

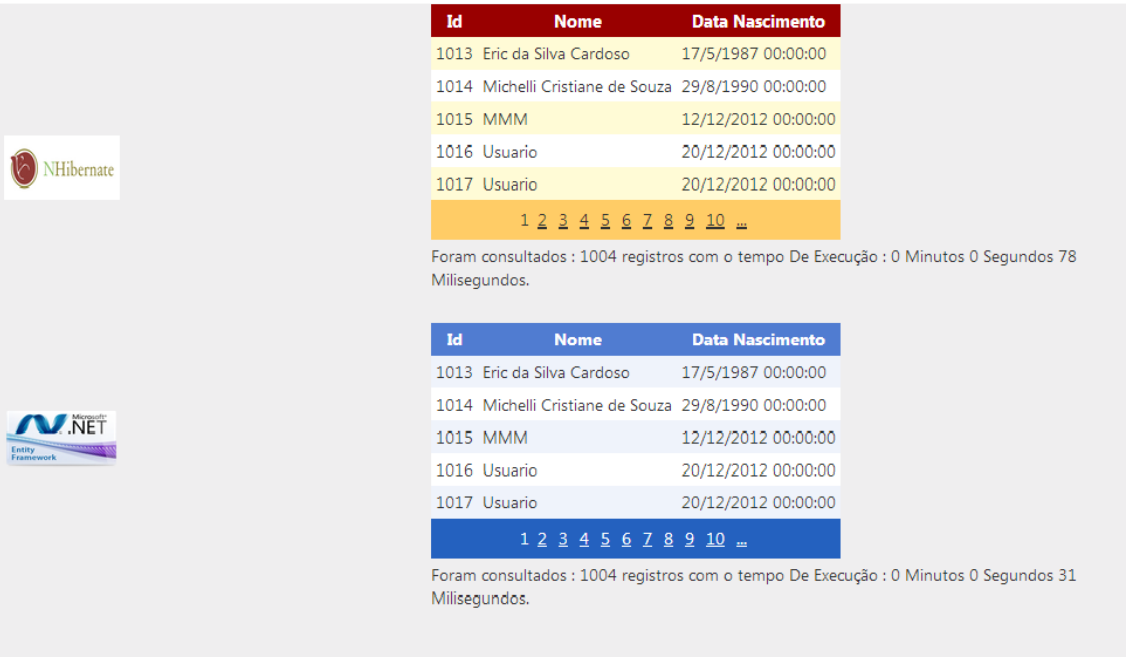
No cadastro do cliente, deverá conter: Nome, sexo, CPF, data de nascimento, e-mail do cliente, escolaridade, situação ocupacional, seus telefones úteis e observações.

No cadastro de alas e quartos por alas, deverá conter: Número ou letra da ALA, número ou letra de cada quarto que esta em uma determinada ALA.

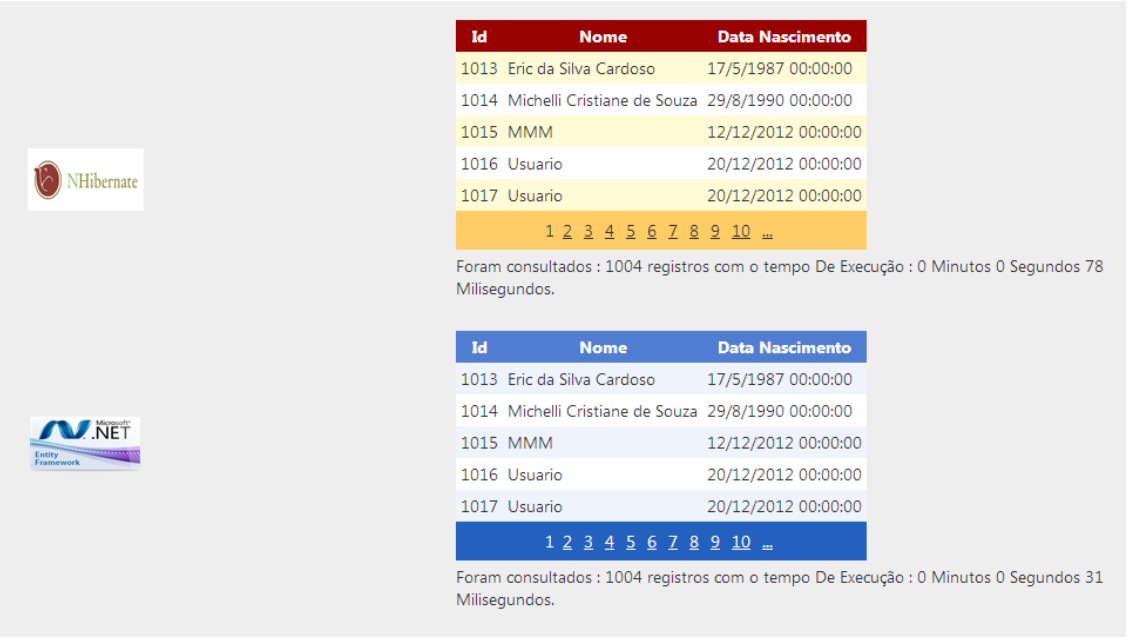
No cadastro do cliente ao condomínio, deverá conter: qual cliente esta entrando, qual quarto de que ALA ele vai ficar, data de entrada, duração de permanência do cliente ao condomínio.

Resultados e discussão:

Teste 1



Teste 2



Como pode-se notar nos testes demonstrados, o desempenho do método de acesso a dados *Entity Framework* obteve um melhor desempenho na consulta em relação ao *NHibernate*, embora a diferença não tenha sido tão grande, uma consulta com maior número de dados poderá notar-se mais satisfatoriamente a diferença de desempenho.

Conclusão:

Com o presente estudo realizado abordando os métodos de acesso a dados *Entity Framework* e o *NHibernate*, pode-se concluir que ambos tem um ótimo rendimento. Porém o *Entity Framework* se sobressai na maioria das operações.

Sendo assim conclui-se que o *Entity Framework* devido a sua agilidade e melhor desempenho de consulta aos dados de um banco de dados deveria ser o método adotado nas empresas de pequeno, médio e grande porte, garantindo assim um menor prazo e custo de seus produtos e dando um melhor desempenho no sistema em relação ao acesso dos dados.