**Porquê Nhibernate?**

Após uma análise às soluções de ORM existentes no mercado duas delas sobressairam, Nhibernate e EF, recaindo a nossa escolha sobre o Nhibernate por várias razões:

Segundo Oren Eini (programador participante no projecto Nhibernate, “tendencioso?!?!”), o NHibernate constitui uma solução mais amadurecida, fruto da sua existência mais duradoura bem como pelo facto de ser uma adaptação do Hibernate para .Net, beneficiar também da aprendizagem adquirida com o desenvolvimento do mesmo. A possibilidade de realização de Batch’s quer de escritas quer de leituras (querys) por forma a realizá-las de uma só vez evitando desta forma um elevado nº de round-trips à BD, para além do esperado carregamento lazy das entidades o Nhibernate possibilita também o Lazy “Extra” que permite por exemplo que a instrução Entity.Count() resulte automáticamente numa instrução SQL “SELECT Count(\*) FROM Entity” evitando desta forma o carregamento de toda a colecção de objecto para memória.

O Nhibernate disponibiliza também vários pontos de extensão contráriamente ao EF existindo já diversos projectos de extensão como são exemplo o Nhibernate Search, Validator ou Shards.

Também o suporte a um elevado número de DB, por exemplo SQL Server, Oracle, Access, PostGres, MySQL entre outras, constiui um ponto a favor da escolha do Nhibernate.

(Oren Eini, A.K.A. Ayende Rahien, equipa Nhibernate <http://ayende.com/Blog/archive/2010/01/05/nhibernate-vs.-entity-framework-4.0.aspx> )

**Porquê XML para o dicionário de dados?**

A adopção da linguagem XML para o dicionário de dados prendeu-se essencialmente com o facto de ser uma linguagem amplamente divulgada e por consequência conhecida da comunidade de programadores (serão eles os utilizadores da solução), a existência de várias API’s para manipulação de documentos XML tornando fácil a navegação no ficheiro, nomeadamente através de expressões XPath, a possibilidade de validação do seu conteúdo através de XML Schema, e a sua transformação através de XSLT.

**Como é que o Nhibernate detecta os objectos dirty?**

Sempre que um objecto é carregado a partir da informação constante na BD, é carregado no Session Cache do Nhibernate, sendo no momento da gravação de objectos comparado com o objecto a gravar (que pode entretanto ter sido alterado) propriedade a propriedade.

**Porquê a escolha de webservices para acesso aos processos de negócio?**

Essencialmente por ser uma tecnologia amplamente divulgada e estudada. ( emais?)

**As questões relacionadas com as chaves das entidades é toda resolvida pelo Nhibernate?**

Não. Por forma a resolver as questões relacionadas com as chaves das entidades foi definido que todas as entidades teriam um campo ID(long/BigInt) o qual garante a unicidade de entidades e consistência da BD. Os campos definidos pelo utilizador como chave da entidade serão definidos como unique na BD e todas as relações entre entidades materializam-se usando as chaves ID’s.

**Que vantagens traz a nossa solução quando comparada com as existentes no mercado?**

É nossa!!

**Porquê a adopção da norma XSD para a validação de tipos?**

Essencialmente por se conseguir especificar todo o domínio de tipos expectável de ser utilizado no âmbito da implementação de uma aplicação.

**Como é feita a implementação/extensão ao nível do controlo de acessos?**

Foi definida a classe abstracta RuntimeRoleProvider que define os métodos necessários à implementação do modelo RBAC2. O RoleProvider a utilizar na aplicação deverá ser definido no dicionário de dados e extender a classe RuntimeRoleProvider sendo chamado de forma declarativa em cada processo de negócio seguindo a nomenclatura de nome de permissão = nome do método.

**Como é garantido que uma nova sincronização não emaga o trabalho entretanto desenvolvido pelo programador/utilizador? (Ou, para que servem as classes denominadas Base Classes?)**

Com a criação das classes Base divide-se o código gerado pela aplicação e o código entretanto desenvolvido pelo utilizador possibilitando desta forma uma nova sincronização.