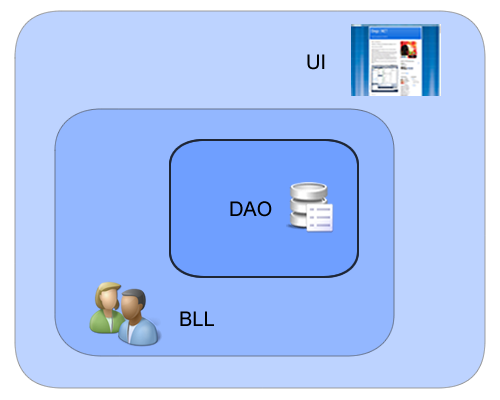
Será um exemplo bem simples, onde o principal objetivo é demostrar desde a criação de vários projetos (camadas) e uma mesma Solution (aplicação) até o relacionamento e comunicação entre as camadas. Para nosso exemplo utilizaremos a metodologia de desenvolvimento em 3 camadas onde temos:

* DAL – Camada de Acesso a dados(Data Access Layer);
* BLL – Camada de Regra de negócios (Business Logic Layer);
* UI – Camada de Apresentação (User Interface);

Segue abaixo uma imagem para ilustração do desenvolvimento em camadas:



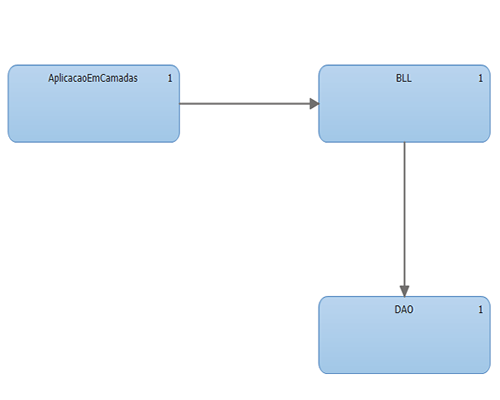
Veja abaixo uma breve descrição sobre cada uma das camadas:

* UI – Camada de Apresentação. Camada responsável por apresentar ao usuário o resultado final de todo o processamento de nosso sistema que se comunica somente com a BLL, não fazendo acesso direto à DAO.
* BLL – Camada de Regra de Negócio. Camada responsável por possuir as regras especifica do sistema a ser desenvolvido, sendo responsável por se comunicar com a DAO (para acessar os dados retornados do banco) e fornecer informações à UI (para repassar o resultado do processamento e configuração das regras do sistema).
* DAO – Camada de Acesso a Dados. Camada responsável se conectar e fazer todas as transações referentes a banco de dados, como consultas, inserções de dados, atualizações de dados e exclusão de dados. Essa camada faz a comunicação com a BL, não se comunicando diretamente com a UI.

**Relacionamento entre camadas**

Já que toda a estrutura de nosso projeto foi criada, temos agora que fazer o relacionamento entre as camadas, para que uma possa ser visível pela outra.

Observe o diagrama abaixo que demostra como será o relacionamento entre elas:



Veja que o diagrama exibe o relacionamento que uma camada possui com a outra. Repare que a nossa camada UI (AplicacaoEmCamada) possui uma relação de dependência com BLL, e BLL possui um relacionamento de dependência com DAO.

Tendo entendido como será o relacionamento entre nossas camadas, vamos fazer os nossos relacionamentos.

Estou desenvolvendo um aplicativo Web típico com as seguintes camadas

1. Camada da interface do usuário (MVC)
2. Camada lógica de negócios (BAL)
3. Camada de acesso a dados (DAL)

Cada camada possui seu próprio objeto DTO, incluindo o BAL e o DAL. Minhas perguntas sobre isso são as seguintes

1. O DTO retornado pelo DAL é simplesmente convertido no DTO correspondente no BAL e enviado à camada da interface do usuário. Os atributos e a estrutura dos objetos DTO são os mesmos em alguns casos. Nesses cenários, é melhor simplesmente retornar o DTO no DAL para a camada da interface do usuário sem incluir um objeto intermediário.
2. Qual é a melhor maneira de nomear esses objetos DTO e outros objetos em cada camada. Devo usar algum prefixo, como DTOName, ServiceName? A razão pela qual estou pedindo para usar um prefixo é porque, se não, as classes na minha solução se chocam com outras classes no Framework e com um prefixo, é mais fácil para mim entender onde cada classe pertence?

[c#](https://qastack.com.br/software/tagged/c%23/)  [design](https://qastack.com.br/software/tagged/design/)  [naming](https://qastack.com.br/software/tagged/naming/)

— [user3631883](https://softwareengineering.stackexchange.com/users/152866/user3631883)  
[fonte](https://softwareengineering.stackexchange.com/questions/259838/naming-conventions-dal-bal-and-ui-layer)

3

[meta.programmers.stackexchange.com/questions/6582/...](http://meta.programmers.stackexchange.com/questions/6582/on-the-troubles-of-naming-and-terminology)

— [mosquito](https://softwareengineering.stackexchange.com/users/31260/gnat)

11

Você está usando espaço para nome?

— [JeffO 13/10](https://softwareengineering.stackexchange.com/users/855/jeffo)

Respostas:

48

**Prefácio**

Espero que isso seja óbvio, mas ... nos espaços de nomes sugeridos abaixo, você substituirá MyCompanye MyProjectpelos nomes reais da sua empresa e projeto.

**DTOs**

Eu recomendaria usar as mesmas classes DTO em todas as camadas. Menos pontos de manutenção dessa maneira. Eu geralmente os coloco em um MyCompany.MyProject.Modelsespaço para nome, em seu próprio projeto VS com o mesmo nome. E eu geralmente os nomeio simplesmente de acordo com a entidade do mundo real que eles representam. (Idealmente, as tabelas do banco de dados também usam os mesmos nomes, mas às vezes faz sentido configurar o esquema de maneira um pouco diferente).

Exemplos: Person, Address,Product

Dependências: nenhuma (exceto as bibliotecas .NET padrão ou auxiliares)

**DAL**

Minha preferência pessoal aqui é usar um conjunto de classes DAL um por um que corresponda às classes DTO, mas em um MyCompany.MyProject.DataAccessespaço para nome / projeto. Os nomes de classe aqui terminam com um Enginesufixo para evitar conflitos. (Se você não gostar desse termo, um DataAccesssufixo também funcionaria bem. Apenas seja consistente com o que você escolher.) Cada classe fornece opções simples de CRUD que atingem o banco de dados, usando as classes DTO para a maioria dos parâmetros de entrada e tipos de retorno (dentro um genérico Listquando houver mais de um, por exemplo, o retorno de um Find()método).

Exemplos: PersonEngine, AddressEngine,ProductEngine

Dependências: MyCompany.MyProject.Models

**BAL / BLL**

Também é um mapeamento um por um aqui, mas em um MyCompany.MyProject.Logicespaço para nome / projeto e com classes recebendo um Logicsufixo. Essa deve ser a **única** camada que chama o DAL! As aulas aqui costumam ser apenas uma passagem simples para o DAL, mas se e quando as regras de negócios precisarem ser implementadas, este é o lugar para isso.

Exemplos: PersonLogic, AddressLogic,ProductLogic

Dependências: MyCompany.MyProject.Models,MyCompany.MyProject.DataAccess

**API**

Se houver uma camada de API de serviços da web, eu uso a mesma abordagem um por um, mas em um MyCompany.MyProject.WebApiespaço para nome / projeto, com Serviceso sufixo da classe. (A menos que você esteja usando a API da Web do ASP.NET, nesse caso, é claro, você usaria o Controllersufixo).

Exemplos: PersonServices, AddressServices,ProductServices

Dependências: MyCompany.MyProject.Models, MyCompany.MyProject.Logic(nunca ignorar isso chamando o DAL directamente!)

**Uma observação sobre lógica de negócios**

Parece ser cada vez mais comum que as pessoas deixem de fora o BAL / BLL e, em vez disso, implementem a lógica de negócios em uma ou mais das outras camadas, onde quer que faça mais sentido. Se você fizer isso, tenha certeza absoluta de que (1) todo o código do aplicativo passa pela (s) camada (s) com a lógica de negócios e (2) é óbvio e / ou bem documentado onde cada regra de negócios específica foi implementada. *Em caso de dúvida, não tente fazer isso em casa.*

**Uma nota final sobre arquitetura de nível corporativo**

Se você estiver em uma empresa grande ou em outra situação em que as mesmas tabelas de banco de dados sejam compartilhadas entre vários aplicativos, recomendo deixar a MyProjectparte fora dos namespaces / projetos acima. Dessa forma, essas camadas podem ser compartilhadas por vários aplicativos front-end (e também por utilitários nos bastidores, como o Windows Services). **Mas faça isso apenas se você tiver uma forte comunicação entre equipes e testes de regressão automatizados completos !!!**Caso contrário, as alterações de uma equipe em um componente principal compartilhado provavelmente quebrarão o aplicativo de outra equipe.