**Estrutura de Projetos**

Sumário

[**1.** **DEFINIÇÕES** 3](#_Toc508880414)

[**2.** **TECNOLOGIA** 3](#_Toc508880415)

[**3.** **ESTRUTURA DA SOLUÇÃO** 5](#_Toc508880416)

[**4.** **ARQUITETURA BACK / FRONT-END** 6](#_Toc508880417)

[Back-End 6](#_Toc508880418)

[**ERP.Modulo.Comum** 7](#_Toc508880419)

[Dependências: 8](#_Toc508880420)

[**ERP.Modulo.Dominio** 8](#_Toc508880421)

[Dependências: 9](#_Toc508880422)

[**ERP.Modulo.Infra** 10](#_Toc508880423)

[Dependências: 10](#_Toc508880424)

1. **DEFINIÇÕES**

Durante o desenvolvimento do projeto, as seguintes terminologias serão utilizadas para identificar os elementos.

|  |  |
| --- | --- |
| ERP | O projeto de ERP como um todo onde todos os módulos se integram e conversam entre si. |
| MODULO | Representa as áreas distintas do ERP a qual as atividades são executadas.  Ex:  RH, Projetos, Comercial, etc. |
| SOLUÇÃO | Um componente do Visual Studio que organizam diversos Projetos (VS) de forma centralizada. |
| PROJETO | Item componente do VS onde será construído as funcionalidades, código, classes, referente a cada elemento do projeto ERP |
| ENTIDADE | No contexto de código, É uma classe que representará algo que seja de identidade consistente no projeto.  Ex: Cliente, Projeto, Atividade  São objetos que serão mapeadas para o banco de dados. |
| CAMADA | E uma separação lógica dos elementos do projeto.  As mais comuns são:   * Camada de Dados * Camada de Domínio * Camada de Apresentação * Camada de Serviços   Há outras camadas sugeridas por algum design patherns, porém não serão abordados |
| DOMINIO | Conjunto de classes e objetos que representam as entidades consistentes do projeto. |

1. **TECNOLOGIA**

Para o desenvolvimento iremos utilizar as seguintes tecnologias abaixo conforme especificações:

|  |  |
| --- | --- |
| Linguagem | C# |
| IDE | Visual Studio 2017 |
| Controle de Versão | TFS online |
| Tipos de Projetos | DLL, Web VMC e Web API |
| Back-End |  |
| * Mapeamento Entidade / Tabela | EntityFramework 6.0 |
| * Método | Code First (Onde o código é criado antes do banco de dados) |
| * Banco de dados | SQL Sever 2012 + |
| Front-End | Web MVC 5 |
| Bootstrap |
| JQuery |

1. **ESTRUTURA DA SOLUÇÃO**

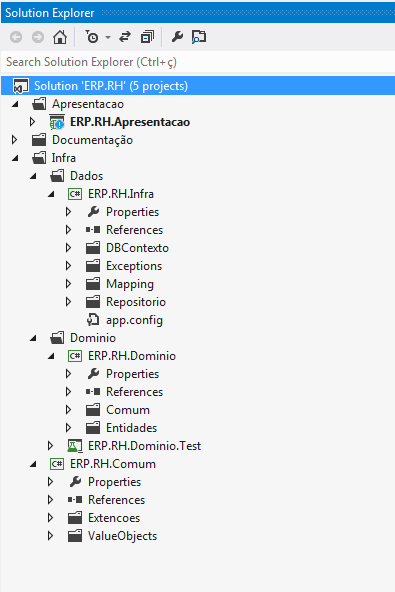
O projeto deverá ser estruturado em diferentes soluções. Cada solução deve integrar os respectivos projetos referente aos módulos.

Abaixo a estrutura básica em que cada projeto deverá ter.

|  |  |
| --- | --- |
| Apresentação | Centraliza projetos referente a saída. |
| Infra |  |
|  |  |
| **Dados** | Contém os projetos que realizam a conexão com o banco de dados |
| **ERP.Modulo.Infra** | Projeto para Infra de dados. Camada de dados de acesso ao banco |
| DBContexto | Onde deverá ser implementadas as classes de contexto do EntityFramwork |
| Exceptions | Classes com as Exceptions de conexao com o banco de dados |
| Mapping | Classes de configurações especificas para mapeamento das entidades com as tabelas no banco de dados. |
| Repósitorio | Implementação do repositório gerenerico das operações CRUD das entidades. |
|  |  |
| **Domínio** | Concentra os projetos no qual estão as entidades de domínio. |
| **ERP.Modulo.Dominio** | Projeto para Infra de dados. Camada de dados de acesso ao banco |
| Entidades | Implementação das classes de entidades do domínio |
| Comum | Classes de uso compartilhado pelo dominio |

Exemplo :

Modulo RH



1. **ARQUITETURA BACK / FRONT-END**

Para cada solução deveremos segmentar os projetos em Back-End e Front-End.

# Back-End

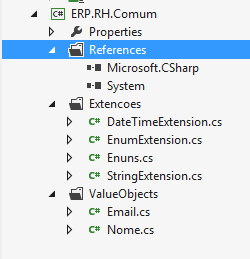
Os projetos de Back-End são :

* **ERP.Modulo.Comum**
* **ERP.Modulo.Dominio**
* **ERP.Modulo.Infra**

Para cada projeto deveremos torna-los com o mínimo de dependência possível.

|  |  |
| --- | --- |
|  | **ERP.Modulo.Comum** |

O primeiro projeto da camada é a de Comum.



Esse projeto deve ter apenas as referências nativas “*Microsoft.CSharp*” e “*System*”.

O propósito desse projeto é conter classes e funcionalidades completamente genéricas e reutilizável por qualquer outro projeto na solução inteira.

Sub pasta:

**🖿 Extensoes**

Concentrar classes estáticas para métodos de extensões que pode ser utilizado por qualquer projeto.

**🖿 ValueObject**

Onde deve ser criada classes ValueObjects as quais são representação de tipos de dados complexos.

Ex:

Cria-se uma classe chamada Nome a qual teria várias propriedades que são relativas ao nome, como Primeiro Nome, Sobrenome, Apelido.

Ao criar uma entidade a qual teria uma propriedade Nome. Ao invés de defini-la como do tipo String, seria definida como no tipo Nome

Public Nome Nome {get, set};

Obs. Essa abordagem não é obrigatória. Apenas orientação de boas práticas.

# Dependências:

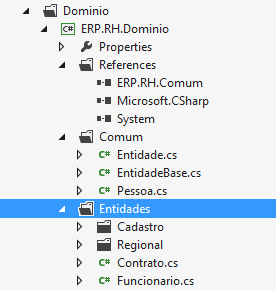
Nenhuma

|  |  |
| --- | --- |
|  | **ERP.Modulo.Dominio** |

O próximo projeto a ser criado é o Domínio.

Esse projeto deve ter a implementação das classes de entidade, sem regra de negócio.

São classes POCO somente com a definição das propriedades.



Estrutura de sub pastas :

**🖿 Comum**

Nessa pasta, deverão estar as classes com implementação genérica e abstratas.

Aqui podem ter também interfaces ou outros tipos de objetos que não serão utilizadas sozinhas. Serão herdadas ou escrita por outras classes de domínio.

**Ex:**

A classe “Pessoa” é uma classe abstrata que poderá representar diversos tipos de entidades distintas com por exemplo:

Um funcionário é uma pessoa

Um usuário do sistema que não necessariamente é uma pessoa, mas pode ser.

Um Cliente, pode ou não ser uma pessoa ou uma empresa.

Um fornecedor também pode ser representado com uma pessoa.

**🖿 Entidades**

Nessa pasta ficam as classes que representam de fato as entidades dos domínios.

As entidades são classes que serão persistidas e mapeadas para tabelas no banco de dados.

A entidade pode ou não derivar de outras classes genéricas como por ex, a classe Pessoa

# Dependências:

*ERP.Modulo.Comum*

|  |  |
| --- | --- |
|  | **ERP.Modulo.Infra** |

O projeto de infra deverá ter a implementação dos acessos ao banco de dados.

Toda a interação com o banco de dados será feita nesse projeto e deverá ser 100% abstrato para as outras camadas

*Sub pastas:*

**🖿 DBContexto**

As classes de contexto ficarão aqui.

Classe de contexto é uma classe especial que herda dá uma Classe DbContext do EntityFramework, que coordena todas as interações com o banco de dados.

É possível definir aqui também algumas configurações de mapeamento global para todos os objetos.

**🖿 Mappings**

Nessa pasta devemos criar as classes de configurações para mapeamento das entidades com as tabelas do banco de dado.

Essas classes podem ser necessárias ou não.

**🖿 Repositorio**

Aqui ficará a implementação das classes para o padrão Repositório onde são definidas as funções básicas CRUD genericamente para todas as entidades.

Caso haja ações especificamente para uma determinada entidade, define-se também uma classe para isso.

# Dependências:

*ERP.Modulo.Dominio*

*ERP.Modulo.Comum*