

# Projeto de Treinamento

Versão do Documento: 1.0

Data: 30/03/2024

---

## Proposta

- Uma aplicação que permitirá a empresa "Nossa Loja" controlar seu dia a dia. Na empresa, precisam ser trabalhados com mais urgência os setores "Cadastro" e "Vendas". Eles funcionam de forma independente, ou seja, o que a equipe de vendas faz não deve interferir no que a equipe de cadastros faz e vice-versa.
- 

## Ambiente de Desenvolvimento

- Visual Studio 2022;
  - .NET 8;
  - MySql 8.0.36;
  - Navicat;
  - GitHub Desktop;
- 

## Arquitetura

- A arquitetura será exatamente a mesma utilizada no SommusGestor.
- Teremos 2 contextos delimitados e um contexto de Core.
  - Contexto **Cadastro**;
  - Contexto **Venda**;
  - Contexto **Core**;
- O acesso ao banco de dados será igual ao utilizado no SommusGestor atualmente.
- A Injeção de Dependência terá o mesmo pacote e implementação do SommusGestor (**Unity**).
- Será desativado em todos os projetos, a verificação de Nulos. Dessa forma a solução será semelhante ao que temos no SommusGestor na versão atual com **.NET Framework**.
- Assim como no SommusGestor, adotaremos apenas testes de integração no projeto. O pacote de testes será o **MSTest**.

---

## Plano de Implementação

- ☐ Criar a Solution;
- ☐ Criar contexto Cadastro;
  - ☐ Projeto Application;
  - ☐ Projeto Application.Test;
  - ☐ Projeto Domain;
  - ☐ Projeto Data;
  - ☐ Projeto Infra;
- ☐ Criar contexto Core;
  - ☐ Projeto Domain;
  - ☐ Projeto Infra;
- ☐ Criar contexto Vendas;
  - ☐ Projeto Application;
  - ☐ Projeto Application.Test;
  - ☐ Projeto Domain;
  - ☐ Projeto Data;
  - ☐ Projeto Infra;
- ☐ Criar contexto Presentation;
  - ☐ Projeto da API;
  - ☐ Projeto do Front-End;

---

## Padrões Adotados no Projeto

### Variáveis

Sempre que possível será utilizado o `var` para declaração das variáveis. Esse padrão faz a inferência de tipo com base no valor recebido.

```
/// ❌ - Não utilizar ou utilizar menos
int idade = 0; // "idade" é do tipo "int"
decimal desconto = 10.5M; // "desconto" é do tipo "decimal"
string texto = "frase"; // "texto" é do tipo "string"
char letra = 'A'; // "letra" é do tipo "char"
bool cancelado = true; // "cancelado" é do tipo "bool"

/// ✅ - Preferência
var idade = 0; // "idade" passa ser do tipo "int"
var desconto = 10.5M; // "desconto" passa ser do tipo "decimal"
var texto = "frase"; // "texto" passa a ser do tipo "string"
```

```
var letra = 'A'; // "letra" passa a ser do tipo "char"
var cancelado = true; // "cancelado" passa a ser do tipo "bool"
```

## Strings

Por padrão será utilizado, para strings vazias, duas aspas duplas no lugar de `string.Empty`.

```
/// ❌ - Não utilizar ou utilizar menos
var nome = string.Empty;

/// ✅ - Preferência
var nome = "";
```

## Funções e Blocos de Código

Sempre que possível, será utilizado a sintaxe simplificada para o corpo de blocos de código. Isso significa que se o bloco possuir apenas uma linha de código dentro dele, o uso das "chaves" será preterido.

```
/// ✅ - Preferência
// Retornar a soma de 1 + 1;
public void SomaUmMaisUm() ⇒ 1 + 1;

/// ✅ - Preferência
// Marcar todos os itens como cancelados
foreach (var item in itens)
    item.Cancelado = false;


/// ✅ - Preferência
// Propriedade protegida contra nulos
private string _nome = "";
public string Nome
{
    get ⇒ _nome ?? "";
    set ⇒ _nome = value;
}
```

## String de Query's do Repositório

Será mantido o formato de texto a seguir para organizar os comandos MySQL escritos. Como são `string`, definir um padrão mínimo ajuda na organização do código. Segue um exemplo de como iremos trabalhar:

```
public cliente Get(int clienteId)
{
```

```

    ///  - Recomendação
    var query = $"
        SELECT cliente_id,
                nome,
                excluido
        FROM cliente
        WHERE cliente_id = ?ClienteId

    ";

    // Restante do código do Método
}

```

Notem que existe uma divisão clara de cada parte do comando, acontecendo justamente nas suas principais palavras chave (no caso "SELECT", "FROM" e "WHERE").

Também é importante se atentar a parte que vai receber as "variáveis" do comando, que são os locais que deixamos para a biblioteca trabalhar a segurança contra os "Sql Injection". No trecho acima passamos `?ClienteId` em Camel Case, diferenciando nitidamente o que é comando do MySql e o que é código da aplicação. O caractere que vamos utilizar para identificação é o `?` antes do nome da variável.

---

## Script do Banco de Dados

Nosso banco de dados ficará no `database` "nossa\_loja". Ele precisa ser criado manualmente antes de rodar o Script abaixo.

Teremos apenas 4 tabelas inicialmente para nossos estudos. Essas tabelas permitirão cobrir todo o cenário inicial da arquitetura. As tabelas em questão são `cliente`, `produto`, `venda`, `venda_item`.

A Seguir está o Script para criar a estrutura do banco de dados.

```

-- _____
-- Estrutura da tabela `cliente`
-- _____

DROP TABLE IF EXISTS `cliente`;
CREATE TABLE `cliente` (
  `cliente_id` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(120) NOT NULL DEFAULT "",
  `cpf` char(11) NOT NULL DEFAULT "",
  `identidade` varchar(15) NOT NULL DEFAULT "",
  `cnpj` char(14) NOT NULL DEFAULT "",
  `ativo` tinyint(1) NOT NULL DEFAULT '0',
  `excluido` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`cliente_id`)
)

```

```

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- _____
-- Estrutura da tabela `produto`
-- _____
DROP TABLE IF EXISTS `produto`;
CREATE TABLE `produto` (
  `produto_id` int(11) NOT NULL AUTO_INCREMENT,
  `descricao` varchar(200) NOT NULL DEFAULT "",
  `preco_custo` decimal(11,4) NOT NULL DEFAULT '0.0000',
  `preco_venda` decimal(11,4) NOT NULL DEFAULT '0.0000',
  `ativo` tinyint(1) NOT NULL DEFAULT '0',
  `excluido` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`produto_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- _____
-- Estrutura da tabela `venda`
-- _____
DROP TABLE IF EXISTS `venda`;
CREATE TABLE `venda` (
  `venda_id` int(11) NOT NULL AUTO_INCREMENT,
  `data_hora` datetime NOT NULL DEFAULT '1000-01-01 00:00:00',
  `cliente_id` int(11) DEFAULT NULL,
  `valor_bruto` decimal(10,4) NOT NULL DEFAULT '0.0000',
  `acrescimo` decimal(10,4) NOT NULL DEFAULT '0.0000',
  `desconto` decimal(10,4) NOT NULL DEFAULT '0.0000',
  `valor_liquido` decimal(10,4) NOT NULL DEFAULT '0.0000',
  `cancelada` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`venda_id`),
  KEY `fk_1_venda` (`cliente_id`),
  CONSTRAINT `fk_1_venda` FOREIGN KEY (`cliente_id`) REFERENCES `cliente`
(`cliente_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- _____
-- Estrutura da tabela `venda_item`
-- _____
DROP TABLE IF EXISTS `venda_item`;
CREATE TABLE `venda_item` (
  `venda_item_id` int(11) NOT NULL AUTO_INCREMENT,
  `venda_id` int(11) NOT NULL DEFAULT '0',
  `numero_item` int(5) NOT NULL DEFAULT '0',
  `quantidade` decimal(11,4) NOT NULL DEFAULT '0.0000',
  `preco_venda` decimal(11,4) NOT NULL DEFAULT '0.0000',
  `excluido` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`venda_item_id`),
  KEY `fk_1_venda_item` (`venda_id`),
  CONSTRAINT `fk_1_venda_item` FOREIGN KEY (`venda_id`) REFERENCES `venda`

```

```
(`venda_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```