Relatório do trabalho da disciplina de Programação Orientada a Objetos

Gestão de Vendedores

Susana Gomes - 18515

Joana Pereira - 23153

Engenharia de Sistemas Informáticos Pós-laboral

Novembro de 2023.

|  |
| --- |
| Afirmo por minha honra que não recebi qualquer apoio não autorizado na realização deste trabalho prático. Afirmo igualmente que não copiei qualquer material de livro, artigo, documento web ou de qualquer outra fonte exceto onde a origem estiver expressamente citada. |

Susana Gomes - 18515

Joana Pereira - 23153

**Índice**

[Introdução 1](#_Toc151233994)

[Estrutura do Projeto 2](#_Toc151233995)

[Aplicação Web ASP.NET Core MVC 2](#_Toc151233996)

[Responsabilidade de cada parte do MVC 3](#_Toc151233997)

[Classes 3](#_Toc151233998)

[Classe SaleStatus 4](#_Toc151233999)

[Classe Department 5](#_Toc151234000)

[Classe SalesRecord 6](#_Toc151234001)

[Classe Seller 7](#_Toc151234002)

[Bibliografia 9](#_Toc151234003)

**Lista de Tabelas**

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

**Lista de Figuras**

[Figura 1: Modelo de domínio. 2](#_Toc151233983)

[Figura 2: Arquitetura Geral. 3](#_Toc151233984)

[Figura 3: Classe SaleStatus. 4](#_Toc151233985)

[Figura 4: Department. 5](file:///C:\Users\Utilizador\Downloads\trabalho_POO_18515_23153_fase1.docx#_Toc151233986)

[Figura 5: Classe SalesRecord. 6](file:///C:\Users\Utilizador\Downloads\trabalho_POO_18515_23153_fase1.docx#_Toc151233987)

[Figura 6: Classe Seller. 7](#_Toc151233988)

# Introdução

O seguinte relatório, tem como objetivo a criação de um projeto em C#, que faça a gestão de vendedores.

Um dos nossos objetivos, é também, a criação de uma interface gráfica entre o utilizador e o nosso programa, de modo a ser mais interativo e facilitada a gestão do mesmo., sendo que nesta primeira fase, apenas fizemos a criação de um projeto Web ASP.NET Core Web Application e implementamos as classes. Na segunda fase do trabalho, esperamos conseguir concluir o mesmo com a estrutura pensada inicialmente.

# Estrutura do Projeto

O projeto foi idealizado do seguinte modo:

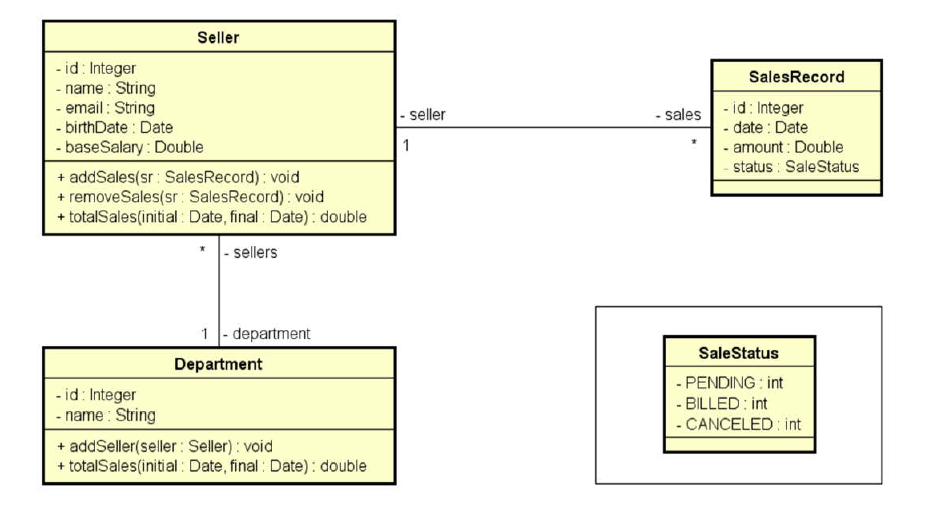


Figura 1: Modelo de domínio.

Um vendedor, terá muitas vendas e vários vendedores estarão associados a um departamento. As vendas terão um estado (SALESTATUS), pendente, faturada ou cancelada.

Basicamente o que pretendemos, é que seja possível adicionar ou remover vendedores, fazer consultas das vendas com base nas datas e no estado em que se encontra a venda, entre outras.

## Aplicação Web ASP.NET Core MVC

O nosso projeto no Visual Studio foi criado com o ASP.NET Core MVC, que é uma estrutura avançada para a criação de aplicações Web, usando o padrão de design Model-View-Controller. É um framework para criação de aplicações web criado pela e comunidade Microsoft, é open source, funciona tanto no .NET Framework quanto no .NET Core, o framework trabalha com uma estrutura bem definida, incluindo:

* Controllers;
* Views;
* Models
  + View Models

### Responsabilidade de cada parte do MVC

**Modelo:** estrutura de entidades de domínio e suas transformações (modelo de domínio).

* Entidades
* Serviços (relacionados a entidades)
* Repositórios (acesso persistente a dados)

**Controladores:** recebem as interações do usuário e tratam as mesmas.

**Visualização:** define a estrutura e o comportamento da interface do usuário.

|  |
| --- |
| Figura 2: Arquitetura Geral. |
| Nesta fase do projeto, ainda estamos a investigar e a tentar perceber como iremos colocar esta parte a funcionar, ficando mais a dizer na próxima e última fase do projeto. Apesar de a ligação ao Mysql ter sido realizada com sucesso, a criação do tema bootstrap também ter sido realizada, entre outras, iremos apenas falar mais sobre o assunto de modo mais profundo na próxima entrega, de modo que, até lá, consigamos aprender mais sobre todas as funcionalidades, comunicação de dados, entre outros, com o frontend e o backend. Ao longo da realização do nosso trabalho, temos utilizado o git, ficando assim o link aqui.  <https://github.com/susanagomes84/Projeto-POO.git> |

## Classes

Nesta primeira fase, fizemos a implementação do modelo de domínio, com os atributos básicos de cada classe. Ficando assim a criação dos métodos para a próxima entrega.

### Classe SaleStatus

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 3: Classe SaleStatus.

Neste caso, usamos esta classe como **enum,** isto porque,vai ser usado para representar os diferentes estados de uma venda.

* **SaleStatusé** o nome da enumeração. Este herda de int, indicando que cada membro enum terá um valor inteiro subjacente.

Os membros individuais do enum são:

* **Pending** com valor 0
* **Billedcom** um valor de 1
* **Canceled** com um valor de 2

### Classe Department

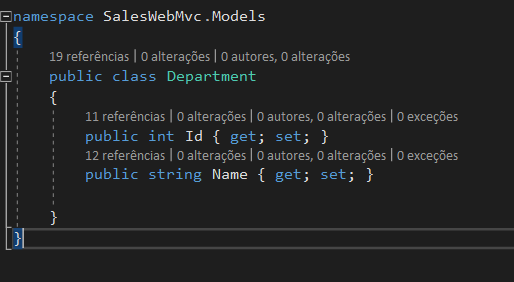


Figura 4: Department.

**Department** é uma classe que representa uma entidade chamada "Departamento". Neste momento apenas tem os atributos associados a ela.

Os atributos associados a ela são:

* **Id:** Uma propriedade do tipo inteiro que representa o identificador único do departamento.
* **Name:** Uma propriedade do tipo string que armazena o nome do departamento.

Estes atributos têm os métodos **get** e **set**, através de propriedades , para obter e atualizar o valor de cada campo que neste caso são públicos.

### Classe SalesRecord

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Figura 5: Classe SalesRecord.

**SalesRecord** é uma classe para representar um registo de venda na nossa aplicação, onde cada instância representa um registo de venda com informações como data, valor, status e o vendedor associado.

Os atributos associados a ela são:

* **Id:** é uma propriedade do tipo inteiro que representa o identificador único do registo de venda.
* **Date:** é uma propriedade do tipo DateTime que representa a data do registo de venda.
* **Amount:** é uma propriedade do tipo double que representa o valor do registo de venda.
* **Status:** é uma propriedade do tipo SaleStatus que representa o status do registo de venda (como "Pending", “Billed” ou “Canceled”).
* **Seller:** é uma propriedade do tipo Seller que representa o vendedor associado a este registo de venda.

### Classe Seller

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 6: Classe Seller.

**Seller** é uma classe onde cada instância representa um vendedor com informações como nome, e-mail, data de nascimento, salário base e departamento associado.

Os atributos associados a ela são:

* **Id:** é uma propriedade do tipo inteiro que representa o identificador único do vendedor.
* **Name:** é uma propriedade do tipo string que representa o nome do vendedor.
* **Email:** é uma propriedade do tipo string que representa o endereço de e-mail do vendedor.
* **BirthDate:** é uma propriedade do tipo DateTime que representa a data de nascimento do vendedor.
* **BaseSalary:** é uma propriedade do tipo double que representa o salário base do vendedor.
* **Department:** é uma propriedade do tipo Department que representa o departamento ao qual o vendedor está associado. Aqui está a relação de composição entre Seller e Department.
* **DepartmentId:** é uma propriedade do tipo inteiro que representa a chave estrangeira (foreign key) para o departamento associado ao vendedor.

# Bibliografia

ardalis. (2022, June 27). Overview of ASP.NET Core MVC. Learn.microsoft.com. https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0

BillWagner. (2023, May 26). Classes - C#. Learn.microsoft.com. https://learn.microsoft.com/pt-pt/dotnet/csharp/fundamentals/types/classes

Violin, R. (2015, February 25). C# – Criando classes, objetos, métodos construtores e acessores. Blog Do Código. https://blogdocodigo.wordpress.com/2015/02/25/c-criando-classes-objetos-metodos-construtores-e-acessores/

C# Properties (Get and Set). (n.d.). Www.w3schools.com. <https://www.w3schools.com/cs/cs_properties.php>

Bootswatch: Free themes for Bootstrap. (n.d.). Bootswatch.com.

https://bootswatch.com/3/

‌