# **Teste de desenvolvimento técnico .NET (C#) - Sistema Informática Empresarial**

Implemente a Api para que seja permitido listar os 5 motoristas mais próximos das coordenadas de latitude e longitude fornecidas na entrada. Apresente o resultado ordenado pela Nota do motorista (da maior para a menor), e de acordo com as seguintes regras:

## Regras de negócio

* Regra 1: Deverá ser implementada uma regra para que não seja permitido realizar a consulta de mais de 5 motoristas por requisição, caso seja informado um número maior que 5 deverá ser lançada uma exceção informando que o número máximo para a consulta é de 5;
* Regra 2: Deverá ser implementada uma regra para quando a nota do motorista for menor que 0 ou maior que 10 lançar uma exceção informando que a nota do mesmo é inválida;
* Regra 3: Deverá ser implementada uma regra para quando o Tipo de Contrato estiver fora da enumeração (descrita no README.md do projeto de domínio) lance uma exceção informando que o mesmo é inválido;

## Projeto

O projeto está dividido nas seguintes camadas:

1 - Camada de apresentação: A camada de apresentação é uma parte essencial do aplicativo. É nesta camada que o usuário final deverá interagir com o sistema, onde está presente as telas para apresentação, consumo de API`s, etc. A camada de apresentação é uma parte essencial do aplicativo. Uma camada de apresentação projetada de forma inadequada pode resultar em complexidade demais, falta de flexibilidade e experiência do usuário ineficiente e frustrante.

2 - Camada de aplicação: A camada de aplicação, é responsável por orquestrar a comunicação entre camadas. Ela serve para que a comunicação entre o que a aplicação recebe de input não seja direta com as camadas inferiores.

3 - Camada de domínio: Esta camada é responsável pela apresentação de conceitos de negócio / domínio, informações sobre o estado dos processos de negócio e implementação de regras, encapsulando toda a lógica de negócio relevante. Basicamente, nesta camada estão presentes classes que implementam a lógica de domínio dentro de seus métodos. Seguindo os padrões de orientação a objetos, esta camada deve ser completamente inconsciente dos detalhes de persistência de dados.

4 - Camada de Infraestrutura (acesso a dados): Esta camada fornece a funcionalidade de acessar os dados. Pode ser dados a partir do sistema em uso ou dados expostos por sistemas externos (Web Services externos, etc.) Assim, essa camada de persistência de dados expõe o acesso a dados para as camadas superiores, normalmente na camada de domínio ou aplicação. Esta exposição deve ser feita de forma dissociada. Em um nível prático, um repositório será normalmente uma classe responsável por executar operações de acesso e de persistência de dados e é, portanto, relacionado a uma tecnologia específica (por exemplo, ligado a um ORM como o Entity Framework, NHibernate, ou mesmo apenas para um ADO.NET, provedor de banco de dados relacional em particular). Ao fazer isso, centralizar a funcionalidade de acesso a dados, torna-se a manutenção e configuração da aplicação mais fácil e mais direta.

5 - Camada de testes: A camada de testes, é responsável por executar os testes automatizados desenvolvidos para a aplicação. Nesta camada estarão presentes todos os níveis de teste, como testes de unidade, testes de integração, testes de performance, entre outros.

Documentar a API com Swagger.

## Modelo

Para o desenvolvimento, deve-se utilizar a abordagem de modelagem de software DDD (Domain Driven Design), seguindo o conjunto de práticas com o objetivo de facilitar a implementação de regras / processos de negócios tratados como domínio.

* Lembre-se: "Toda arquitetura é Design, mas nem todo Design é arquitetura" - Grady Booch.
* É importante seguir as convenções de codificação em C#.
* Cada projeto contém um arquivo README.md com detalhes da implementação que deve ser feita.