## Design structure

## A estrutura do Sistema é cliente server. Haverá um servidor para a aplicação e banco de dados. O código cliente roda em um browser que tenha suporte a html e javascript. O host da aplicação é feito através do Microsoft IIS, enquanto o banco de dados é o Microsoft Sql Server.

## Alto nível de como funciona a execução da aplicação:



E aqui um high level de como o sistema está estruturado:

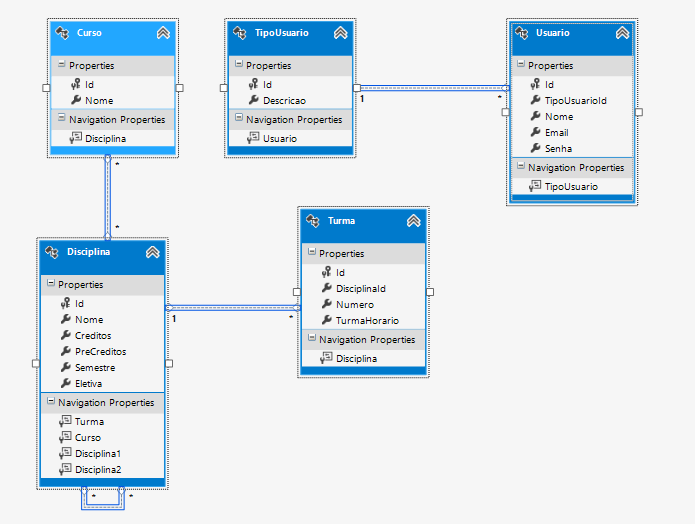


Existem basicamente 3 camadas de sistema. Presentation, que é a camada de apresentação, onde reside a maior parte do código cliente. Business Logic, que é onde há a lógica da aplicação, as regras de negócio. E a Data Access Layer, que é a camada de persistência de dados.

## Subsystems

### [Data Access Layer]

A camada de acesso aos bancos possui os repositórios que serão usados para acesso aos dados do sistema, como Turma, Disciplina, etc...

ER:  


### [Business Logic Layer]

Está camada possui o OracleService, responsável por fazer a lógica de sugerir as disciplinas. Ainda, aqui deverá ser adicionado qualquer outra classe responsável por realizar lógica de negócio. Essa camada é o core da aplicação.

Ela possui três projetos:   
AcadOracle.DomainModels: contém os models do sistema.  
AcadOracle.Core: possui funcionalidades de infraestrutura, como logging, error handling, etc.  
AcadOracle.BusinessLogic: camada com as regras de negócio do sistema.

UML DomainModels:



### [Presentation Layer]

É a camada que possui o código Web MVC. Utiliza o padrão MVC com a tecnologia ASP.NET MVC. Essa camada contém a lógica de apresentação do sistema para o usuário. Todas as views e controllers estão nessa camada. CRUDs e tela de Sugestão de Disciplinas moram na presentation Layer.

## Patterns

### [Repository]

#### Overview

O pattern de repositório é excelente pois abstrai a camada de persistência do cliente, e ainda permite uma forma prática de consulta dos dados, no formato repositório:



#### Structure

Qualquer parte do sistema pode declarar referenciar uma interface repositório para consultar os dados.

#### Behavior

Esse pattern irá abstrair o acesso ao banco. Nesse caso, sua implementação terá acesso direto às tabelas do entity framework, o ORM utilizado.

Example

## 

### [MVC]

#### Overview

Model View Controller, é o pattern utilizado para delegar corretamente as responsabilidades do sistema. A lógica da tela é gerida pelo controller, as Views serão renderizadas pelo controller e os models são os modelos utilizados na view.



#### Structure

O MVC está no projeto AcadOracle.PresentationLayer.

#### Behavior

## O acesso web chega no controller que renderiza a view e executa as ações necessárias para conseguir os dados necessários para a view. O Controller ainda transforma as estruturas (models) do DomainModel em ViewModels.

## Requirement realizations

### [Sugestão de disciplinas]

#### View of participants

Abaixo segue um diagrama de como deve funcionar o processo de sugestão de disciplinas:



### [CRUD Turma]

#### Criar



#### Editar



#### Deletar

