

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

INSTITUTO POLITÉCNICO DA GUARDA

# Gestão de Horários das Provas

R E L ATÓ R I O D O T R A B A L H O P R ÁT I C O

|  |
| --- |
| **Curso**  **Unidade Curricular**  **Ano Letivo**  **Docente**  **Data**  **Alunos** |

Engenharia Informática Engenharia de Software II   
2018/2019   
Maria Clara Silveira  
15/09/2018   
Fábio Martins nº1012502

Conteúdo

[Gestão de Horários das Provas 1](#_Toc535500170)

[Introdução 4](#_Toc535500171)

[Descrição do tema do projeto 5](#_Toc535500172)

[Diagrama de Contexto 6](#_Toc535500173)

[Padrões de Desenvolvimento 8](#_Toc535500174)

[Estado de Arte 9](#_Toc535500175)

[Tabela de atores, objetivos e respetivos casos de uso 10](#_Toc535500176)

[Diagrama de Casos de Uso 11](#_Toc535500177)

[Descrição de Casos de Uso 13](#_Toc535500178)

[Diagrama de Sequência dos Casos de Uso dentro da fronteira 14](#_Toc535500179)

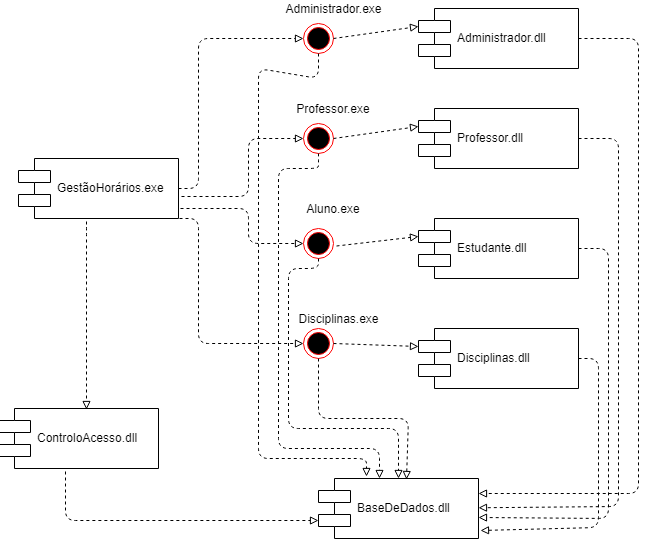
[Diagrama de Classes 15](#_Toc535500180)

[Diagrama de Estados 16](#_Toc535500181)

[Diagrama de Pacotes 17](#_Toc535500182)

[Diagrama de Instalação 19](#_Toc535500183)

[Diagrama de Componentes 20](#_Toc535500184)

[ 20](#_Toc535500185)

[Diagrama de Atividades 22](#_Toc535500186)

[Semântica de Classes 23](#_Toc535500187)

[Classe Professor 23](#_Toc535500188)

[Classe Estudante 24](#_Toc535500189)

[Classe Horario 25](#_Toc535500190)

[Classe Disciplina 26](#_Toc535500191)

[Conclusão 27](#_Toc535500192)

[Anexos 28](#_Toc535500193)

## Introdução

Este projeto assenta em ajudar na elaboração dos horários para as frequências, exames, exames de recurso e época especial do curso de Engenharia Informática. Pretende assim ser uma ajuda tanto para os professores como para o administrador, a marcação deste tipo de avaliações para que não haja sobreposição de provas.

O projeto é concebido no ceio de duas unidades curriculares, Engenharia de Software II e Programação para a Internet, onde em concordância uma com a outra, estudamos o desenvolvimento do software em si, desde os métodos como se desenvolve, especificamente o método de desenvolvimento Ágil ate a elaboração de diversos casos de uso que apos serem aplicados os vários padrões de desenvolvimento de casos de uso resulta uma otimização das tarefas a serem concebidas num website desenvolvido com o modelo MVC em ASP.NET Core.

## Descrição do tema do projeto

### Neste projeto inicialmente irá existir autenticação, sendo atribuídas os seguintes estatutos: Administrador, Professor, Aluno.

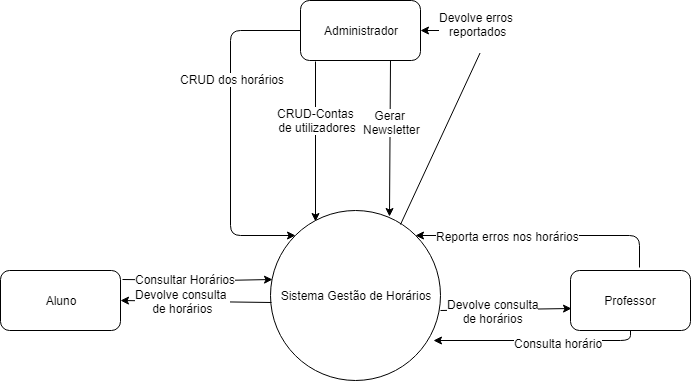
### O aluno entra no site e poderá ver os horários do curso de informática, após login com sucesso. Caso o aluno efetue o login, poderá editar o seu perfil e terá acesso às disciplinas em que se encontra matriculado e informações sobre as datas das avaliações.

### O professor após efetuar o login, pode editar o seu perfil assim como pode também marcar avaliações às disciplinas no qual é docente.

### O administrador, pode gerir todas as cadeiras existentes no curso, gerir as provas de avaliação e controlar os possíveis erros das sobreposições das avaliações, seja por causa do ano das respetivas cadeiras, seja por causa do horário, resumindo, poderá gerir de forma automática todas as épocas (época de frequências, exames, exames de recurso e época especial).

## Diagrama de Contexto

O diagrama de contexto é composto por fluxos de dados( que simbolizam interfaces) entre o sistema e entidades externas, permitindo assi, identificar os limites dos processos, as áreas envolvidas com o processo, os relacionamentos com outros processos, elementos externos representando assim o sistema como um único processo. O diagrama que se segue ilustra o diagrama de contexto que traduz a solução para o problema apresentado. Neste caso como entidades externas temos o Administrador, Professor e Aluno.



## Padrões de Desenvolvimento

Two TierReview

#### Hoje em dia, com a complexidade dos projetos, a necessidade de rever todos os casos de uso é enorme, pois trará benefícios económicos como a diminuição do tempo de execução.

#### As revisões ao longo de um projeto ajudam a validar, verificar e avaliar todo o processo de desenvolvimento.

#### Estas revisões não podem ser feitas por quem as executa, pois é muito difícil para os developers encontrarem erros no seu trabalho, pois estão muito familiarizados com todo o trabalho que desenvolveram. Portanto esta revisão é melhor ser feita por quem nunca olhou para aquilo, pois será mais fácil detetar erros e inconsistências.

#### Este padrão de caso de uso explica assim deverá existir revisões ao longo do projeto e que devem ser divididas por dois tipos de revisão:

#### -Várias revisões ao longo do projeto, com uma equipa pequena (3 a 4 elementos) e interna;

#### -Pelo menos uma revisão em que estão presentes toda a equipa envolvida no projeto, de modo a verificar todos os casos de uso, para se saber se estão todos bem descritos e fáceis de perceber, se são fáceis de implementar e se são precisos.

#### Depois desta revisão, se tudo estiver bem, então a equipa chegarão à conclusão que se encontra em QuittingTime(se os casos de uso responderem ás três questões fundamentais deste padrão).

#### 

## Estado de Arte

## Tabela de atores, objetivos e respetivos casos de uso

|  |  |  |
| --- | --- | --- |
| **Ator** | **Casos de Uso** | **Objetivos** |
| Administrador | Criar Professor | O administrador cria uma conta para cada professor |
| Modificar professor | O administrador pode alterar qualquer informação de um professor |
| Inativar Professor | o administrador pode inativar uma conta de um professor |
| Consultar Professor | O administrador pode consultar a conta de qualquer professor |
| Criar Aluno | O administrador cria uma conta para cada aluno |
| Modificar Aluno | O administrador pode modificar uma conta de aluno |
| Inativar Aluno | O administrador pode inativar uma conta de um aluno |
| Consultar Aluno | O administrador pode consultar uma conta de um aluno |
| Gerar Newsletter | O administrador gera uma newsletter e envia para todos os alunos/professores |
| Criar horário | O administrador pode adicionar disciplinas ao horário |
| Modificar horário | O administrador pode apagar disciplinas do horário |
| Professor | Consultar Conta | O professor pode consultar a sua conta |
| Consultar Horários | O professor pode consultar o horário |
| Reportar um Erro/Aviso | O professor pode reportar erros sobre o horário ao administrador |
| Aluno | Consultar Conta | O aluno pode consultar a sua conta |
| Consultar horário | O aluno pode consultar horário |

## Diagrama de Casos de Uso

Diagrama de casos candidatos e dentro da fronteira

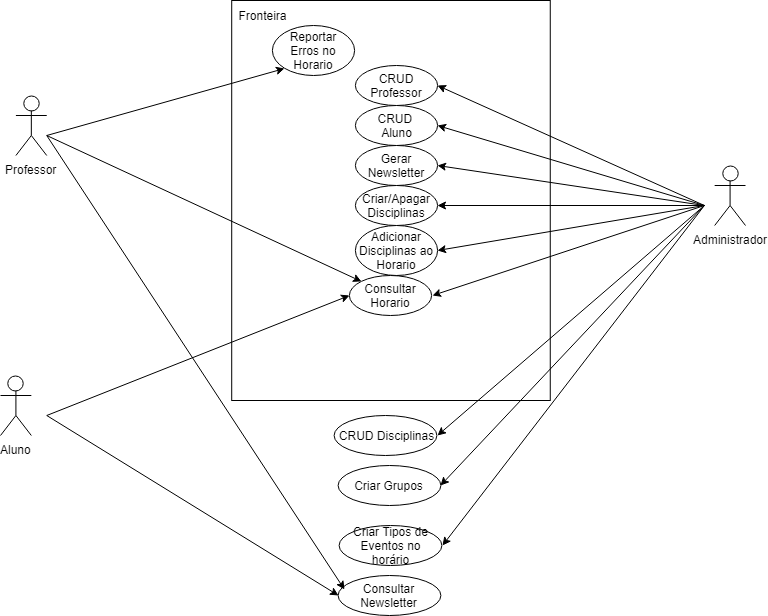
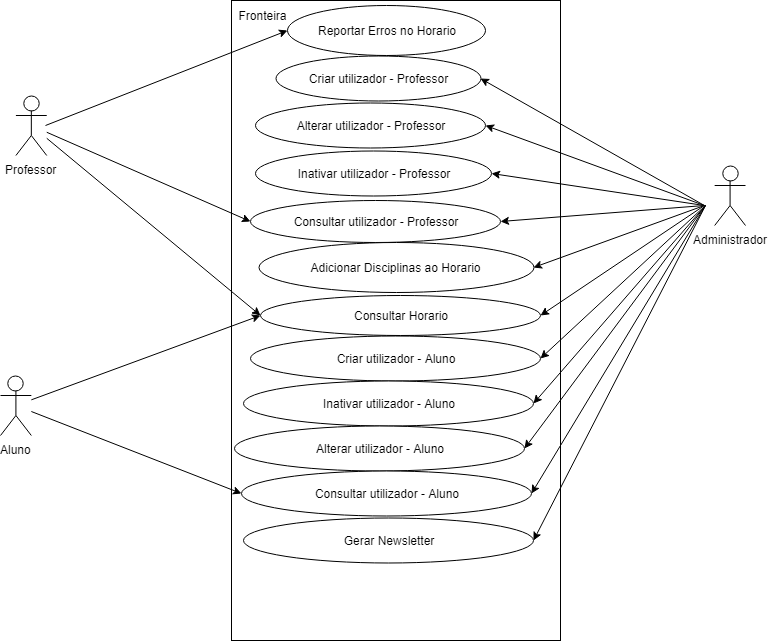


Diagrama de casos dentro da fronteira



## Descrição de Casos de Uso

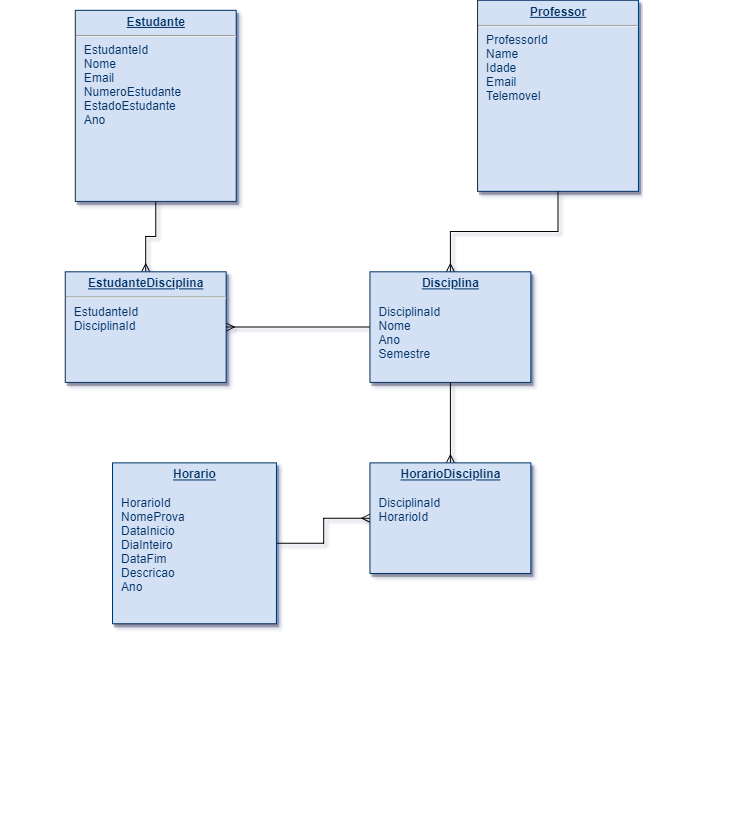
|  |  |
| --- | --- |
| Nome do caso de uso | RF01, Autenticar Ator (Login) |
| Descrição do requisito | O objetivo deste caso de uso é controlar o acesso ao sistema de software |
| Tamanho | 21 |
| Pré-condição |  |
| Caminho Principal |  |
|  | 1-Acede ao software.  2-Pede a sua identificação (login e password).  3-Introduz login e password.  4-Apresenta o horário com os eventos. |
| Caminhos Alternativos | 3a-Se o email de utilizador estiver errado o sistema volta a pedia a identificação.  3b-Se a password estiver incorreta o sistema volta a pedir a sua password. |
| Suplementos ou Adornos |  |

|  |  |
| --- | --- |
| Nome do caso de uso | RF02, Adicionar prova no horário |
| Descrição do requisito | O objetivo deste caso de uso é a criação de eventos(provas). |
| Tamanho | 21 |
| Pré-condição | Ser admin e estar com login válido. |
| Caminho Principal |  |
|  | 1-O Administrador(ator) seleciona a opção “Marcação”  2-O sistema abre uma página com um calendário  3-O ator clica no dia em que pretende marcar a prova  4-O sistema devolve um formulário para criar evento  5-O ator preenche e clica em “Guardar”  6-O sistema cria esse evento e adiciona á tabela horário com um ID  7-É mostrado uma mensagem de sucesso |
| Caminhos Alternativos | 5a-se algum dos campos preenchidos não corresponder á sua validação, é exibida uma mensagem de erro |
| Suplementos ou Adornos |  |

|  |  |
| --- | --- |
| Nome do caso de uso | RF03, Criar User |
| Descrição do requisito | O objetivo deste caso de uso é criar novo User |
| Tamanho | 8 |
| Pré-condição | O Administrador tem o login válido. |
| Caminho Principal |  |
|  | 1-O diretor seleciona a opção “Criar Estudante”  2-O sistema abre o formulário com os dados a introduzir  3-O administrador insere os atributos nome, email, numero de estudante ano, estado do estudante  4-O administrador carrega na opção guardar  5-O sistema atribui automaticamente o ID ao novo estudante.  6-O sistema dá sucesso. |
| Caminhos Alternativos | 2a-O sistema tem opção de voltar à lista  3a-Se o atributo nome estiver nulo o sistema pede para inserir um nome.  3b- Se o atributo nome estiver incorreto o sistema pede para inserir um nome válido  3c- Se o atributo email estiver nulo o sistema pede para inserir um email.  3d- Se o atributo email estiver incorreto o sistema pede para inserir um email válido. |
| Suplementos ou Adornos | * Testar se o sistema deixa registar estudantes sem os campos obrigatórios estarem preenchidos.   Teste1-Validar email  Utilizador insere no Email = asdasdadsa  O sistema avisa que email não é válido.   * Teste2– Validação do Nome   Utilizador insere – Nome = Fabio12  O sistema notifica – O Nome apenas pode conter caracteres. |

## Diagrama de Sequência dos Casos de Uso dentro da fronteira

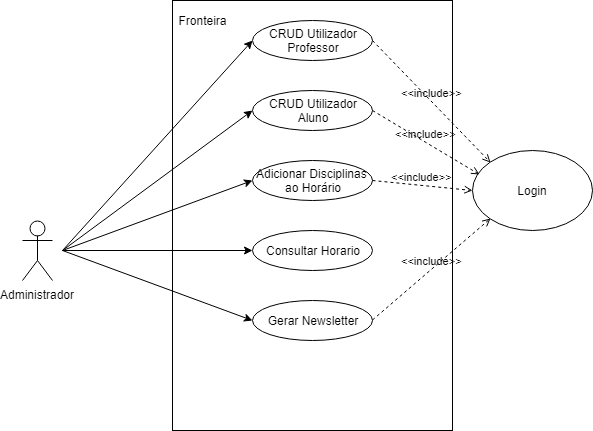
## Diagrama de Classes

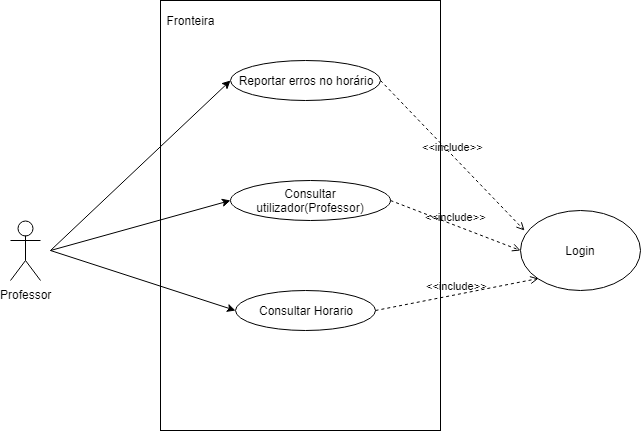


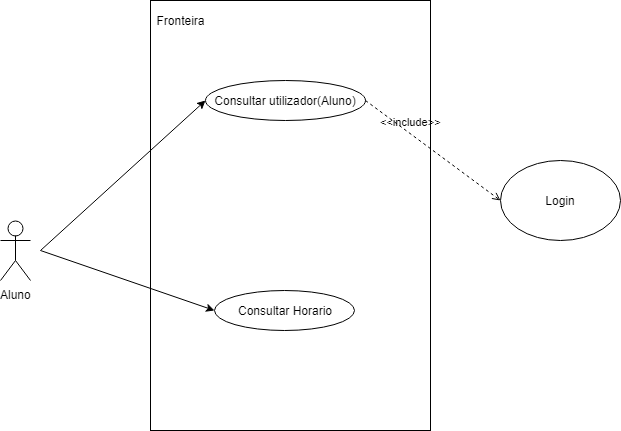
## Diagrama de Estados

## Diagrama de Pacotes

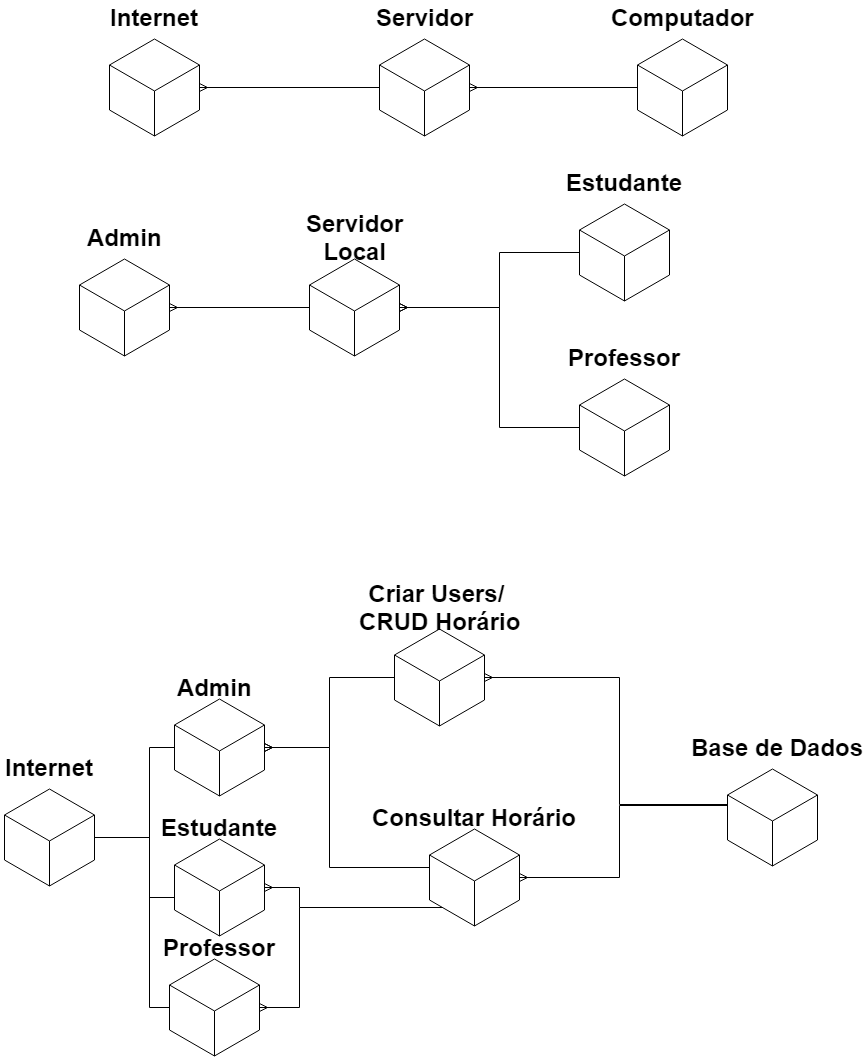
Diagrama de Pacotes-Atores







## Diagrama de Instalação



## Diagrama de Componentes

## 

**Descrição:**

Professor.exe: Responsável pelas operações relacionadas com o Professor(criar, alterar, apagar, consultar). Depende do componente Professor.dll pois necessita de informação acerca do Professor. Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende da BaseDados.dll, pois necessita de guardar a sua informação numa base de dados.

Estudante.exe: Responsável pelas operações relacionadas com o Estudante(criar, alterar, apagar, consultar). Depende do componente Estudante.dll pois necessita de informação acerca do Estudante. Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende da BaseDados.dll, pois necessita de guardar a sua informação numa base de dados.

Administrador.exe: Responsável pelas operações relacionadas com o Administrador(criar, alterar, apagar, consultar). Depende do componente Administrador.dll pois necessita de informação acerca do Administrador. Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende da BaseDados.dll, pois necessita de guardar a sua informação numa base de dados.

Disciplina.exe: Responsável pelas operações relacionadas com o Disciplina(criar, apagar, consultar). Depende do componente Disciplina.dll pois necessita de informação acerca das Disciplinas. Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende da BaseDados.dll, pois necessita de guardar a sua informação numa base de dados.

## Diagrama de Atividades

## Semântica de Classes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe Professor | | | | | | |
| Nome do Campo | **Tipo de Dados** | **Descrição** | **Valores válidos** | **Formato** | **Restrições** | **Valor por defeito** |
| ProfessorId | INT | Número sequencial que identifica o professor | >0 | Até 2147483648 caracteres | Só pode ser gerado pelo sistema e não pode ser alterados | - |
| Nome | String | Nome do professor | Carateres de A a Z | Entre 2 a 80 Carateres | Obrigatório/ Alterável | - |
| Idade | INT | Idade do professor | >0 | Até 2147483648 caracteres | - | - |
| Email | String | Email do professor | Todo o tipo de carateres com a presença obrigatória de um @ | - | Obrigatório | - |
| Telefone | INT | Telefone de contato do professor | Valores entre 0 e 9 | Até 9 caracteres | Obrigatório / Alterável | - |
| Operações | | | | | | |
| Nome | **Descrição** | | | | | |
| Consultar() | Permite operações como Consultar() da classe Horario. | | | | | |
| Criar() | Operação para criar um “aviso” no horário | | | | | |
| Editar() | Operação de editar dados do professor | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe Estudante | | | | | | |
| Nome do Campo | **Tipo de Dados** | **Descrição** | **Valores válidos** | **Formato** | **Restrições** | **Valor por defeito** |
| EstudanteId | INT | Número sequencial que identifica o professor | >0 | Até 2147483648 caracteres | Só pode ser gerado pelo sistema e não pode ser alterados | - |
| Nome | String | Nome do professor | Carateres de A a Z | Entre 2 a 80 Carateres | Obrigatório/ Alterável | - |
| NumeroEstudante | INT | Numero do estudante | >0 | Até 2147483648 caracteres | Obrigatório | - |
| EstadoEstudante | Booleano | Saber estado do estudante | True/false | - | Obrigatório | - |
| Ano | INT | Ano do aluno | Valores entre 1 e 3 | Um digito | Obrigatório / Alterável | - |
| Operações | | | | | | |
| Nome | **Descrição** | | | | | |
| Consultar() | Permite que operações como consultar() da classe horário e da classe Estudante estejam disponiveis | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe Horario | | | | | | |
| Nome do Campo | **Tipo de Dados** | **Descrição** | **Valores válidos** | **Formato** | **Restrições** | **Valor por defeito** |
| HorarioId | INT | Número sequencial que identifica o professor | >0 | Até 2147483648 caracteres | Só pode ser gerado pelo sistema e não pode ser alterados | - |
| NomeProva | String | Nome da prova | Carateres de A a Z | Entre 2 a 80 Carateres | Obrigatório/ Alterável | - |
| DataInicio | Date | Data Inicio da prova | - | - | Obrigatório | - |
| DiaInteiro | Booleano | Saber se a prova é o dia inteiro | True/false | - | Obrigatório | - |
| DataFim | Date | Data fim da prova | - | - | Obrigatório / Alterável | - |
| Operações | | | | | | |
| Nome | **Descrição** | | | | | |
| Consultar() | Permite que operações como Criar(), Editar() da classe horário consultem as avaliações disponíveis | | | | | |
| Criar() | Operação que permite a criação de uma prova no horário | | | | | |
| Editar() | Operação de edição de uma prova no horário | | | | | |
| Desativar() | Permite remover uma prova do horário | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classe Disciplina | | | | | | |
| Nome do Campo | **Tipo de Dados** | **Descrição** | **Valores válidos** | **Formato** | **Restrições** | **Valor por defeito** |
| DisciplinaId | INT | Número sequencial que identifica o professor | >0 | Até 2147483648 caracteres | Só pode ser gerado pelo sistema e não pode ser alterados | - |
| Nome | String | Nome da Disciplina | Carateres de A a Z | Entre 2 a 80 Carateres | Obrigatório/ Alterável | - |
| Ano | INT | Ano da Disciplina | >0 | Até 2147483648 caracteres | - | - |
| Operações | | | | | | |
| Nome | **Descrição** | | | | | |
| Consultar() | Permite que operações como Criar(), Editar() da classe Horário consultem as provas disponíveis | | | | | |
| Criar() | Operação que permite a criação de uma Disciplina | | | | | |
| Editar() | Operação de edição de Uma disciplina | | | | | |
| Desativar() | Permite que a Disciplina seja removida. | | | | | |

## Conclusão

## Anexos

Diário de Atividades

Prototipagem

Algoritmo do Digito de Controlo

#### Bilhete de Identidade e Cartão do Cidadão

#### MODO DE CÁLCULO:

#### Se o número tiver só 6 ou 7 dígitos acrescente 0 (zeros) à esquerda até perfazer 8 dígitos

#### MULTIPLIQUE O

#### 

#### 8.º dígito por 2

#### 7.º dígito por 3

#### 6.º dígito por 4

#### 5.º dígito por 5

#### 4.º dígito por 6

#### 3.º dígito por 7

#### 2.º digito por 8

#### 1.º dígito por 9

#### 

#### Adicione os resultados

#### Calcule o Módulo 11 do resultado, isto é, o resto da divisão do número por 11.

#### Se o resto for 0 ou 1, o dígito de controle será 0

#### Se for outro algarismo x, o dígito de controle será o resultado de 11 - x

#### public bool ValidateNumeroDocumentoCC(string numeroDocumento) { int sum = 0; bool secondDigit = false;

#### 

#### if(numeroDocumento.Length != 12) throw new ArgumentException("Tamanho inválido para número de documento.");

#### for (int i = numeroDocumento.Length-1; i >= 0; --i) { int valor = GetNumberFromChar(numeroDocumento[i]);

#### 

#### if (secondDigit) { valor \*= 2;

#### 

#### if (valor > 9) valor -= 9; }

#### 

#### sum += valor; secondDigit = !secondDigit;

#### }

#### 

#### return (sum % 10) == 0; }

#### 

#### public int GetNumberFromChar(char letter) { switch(letter) { case '0' : return 0; case '1' : return 1;

#### case '2' : return 2;

#### case '3' : return 3;

#### case '4' : return 4;

#### case '5' : return 5;

#### case '6' : return 6;

#### case '7' : return 7;

#### case '8' : return 8;

#### case '9' : return 9;

#### case 'A' : return 10;

#### case 'B' : return 11;

#### case 'C' : return 12;

#### case 'D' : return 13;

#### case 'E' : return 14;

#### case 'F' : return 15;

#### case 'G' : return 16;

#### case 'H' : return 17;

#### case 'I' : return 18;

#### case 'J' : return 19;

#### case 'K' : return 20;

#### case 'L' : return 21;

#### case 'M' : return 22;

#### case 'N' : return 23;

#### case 'O' : return 24;

#### case 'P' : return 25;

#### case 'Q' : return 26;

#### case 'R' : return 27;

#### case 'S' : return 28;

#### case 'T' : return 29;

#### case 'U' : return 30;

#### case 'V' : return 31;

#### case 'W' : return 32;

#### case 'X' : return 33;

#### case 'Y' : return 34;

#### case 'Z' : return 35;

#### }

#### 

#### throw new ArgumentException("Valor inválido no número de documento.");