Instituto Politécnico da Maia

Provas Públicas para atribuição do Título de Especialista

Projeto "Pautas Prontas"

APLICAÇÃO WEB DE GESTÃO DE PAUTAS DE AVALIAÇÃO

José Manuel Boturão das Neves

ISTEC

2022

Instituto Politécnico da Maia

Provas Públicas para atribuição do Título de Especialista

CNAEF 481 - Ciências Informáticas

Portaria n.º 256/2005, de 16 de março

Projeto "Pautas Prontas"

APLICAÇÃO WEB DE GESTÃO DE PAUTAS DE AVALIAÇÃO

José Manuel Boturão das Neves

ISTEC

2022

1 PREFÁCIO DO AUTOR

O autor desde há cerca de 30 anos tem vindo a desenvolver a atividade enquanto docente em diversos âmbitos, tais como:

- Algoritmia e estrutura de dados;
- Linguagens de programação tais como: Linguagem C, C++, c#, PHP, JAVA, JAVASCRIPT e PYTHON;
- Tem ainda lecionado nas áreas do desenvolvimento de "*desktop applications* [1]" (*Windows Presentation Foundation*), aplicações WEB *(PHP, ASP.NET e Node Js*) e ainda no desenvolvimento de aplicações móveis (*Android Studio*).
- Estrutura e arquitetura de base de dados;
- Sistemas de gestão de base de dados.

Tem complementado esta atividade pedagógica com a colaboração no desenvolvimento de aplicações para algumas empresas e instituições, com destaque para a Rede de Emissores Portugueses e a Academia de Software do Instituto Superior de Tecnologias Avançadas de Lisboa.

Relativamente às suas habilitações académicas obteve o grau académico de licenciatura concluindo o curso superior de Informática ministrado no ISTEC.

[1] A desktop application is a software program that can be run on a standalone computer to perform a specific task by an end-user.

What is a Desktop App? (n.d.). V2 Cloud. https://v2cloud.com/glossary/what-is-a-desktop-app

Desde sempre manifestou apetência pelas áreas de programação e sistemas de gestão de base de dados e domina, neste contexto, as seguintes plataformas de desenvolvimento:

- Asp.NET – Desenvolvimento de aplicações WEB;
- XAMP – Desenvolvimento de aplicações WEB
- Android Studio – Desenvolvimento de aplicações móveis.
- Unity – No desenvolvimento de jogo em 3D e 2D.

Relativamente às linguagens de programação domina:

- C++;
- C#;
- JAVA;
- JAVASCRIPT;
- PHP;
- Node JS e
- PYTHON.

No que diz respeito a estruturas e sistemas de gestão de Base de dados tem bons conhecimentos de *SQL SERVER* e MYSQL no contexto das bases de dados relacionais e de "*MongoDB*" no contexto de bases de dados não relacionais.

O seu endereço eletrónico é o seguinte: *boturao@netcabo.pt*

2 RESUMO

No contexto da docência num estabelecimento de ensino superior, verificou-se a necessidade de se agilizar e facilitar o processo de gestão de pautas tornando possível que a consulta e o registo das avaliações escolares pudessem ser efetuados em modo "*on-line*".

Nesta conformidade, procedeu-se ao desenvolvimento de uma aplicação *WEB* recorrendo-se à utilização de diversas tecnologias, nomeadamente o ASP.NET e o sistema de gestão de base de dados SQL-SERVER.

Concluída a aplicação, verificou-se que a maioria dos requisitos foram cumpridos passando a elencar-se os mais relevantes:

- Os serviços da secretaria escolar adquirirem a capacidade de emitir pautas decorrentes das avaliações escolares;
- Os professores terem a capacidade de registo as suas avaliações em modo on-line;
- Os alunos conseguirem consultar as pautas através da aplicação WEB;
- Todos estes processos estarem assegurados por um sistema de controlo de permissões e de autenticação fiável;

No final do projeto vislumbrou-se a necessidade de se emitirem certificados de habilitações em formato "*pdf*". Todavia, este desiderato ainda não foi implementado.

3 ABSTRACT

In the context of teaching in a higher education institution, there was a need to streamline and facilitate the process of managing guidelines, making it possible for consultation and recording of school evaluations to be carried out in an "online" mode.

Accordingly, the developing of a WEB application using the latest technological tools available, more precisely, the use of ASP.NET technology and the SQL-SERVER database management system.

After completing the application, it was found that most of the requirements were met, and the most important ones were listed:

- The services of the school secretariat have the ability to issue guidelines based on the curricula being evaluated.
- The professors can register on-line their evaluations.
- The students can consult the evaluation results.
- All these processes are safeguarded by reliable authentication and security policies.

In the end of the project, it was foreseen the necessity of issuing the certificate, something yet to be improved.

4 ÍNDICE

5 ÍNDICE DE FIGURAS

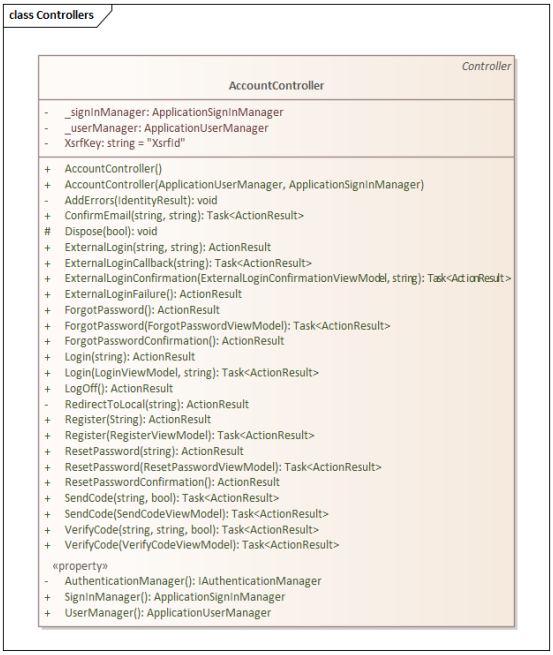
6 ÍNDICE DE CASOS DE USO

Casos

7 DICIONÁRIO DE CLASSES

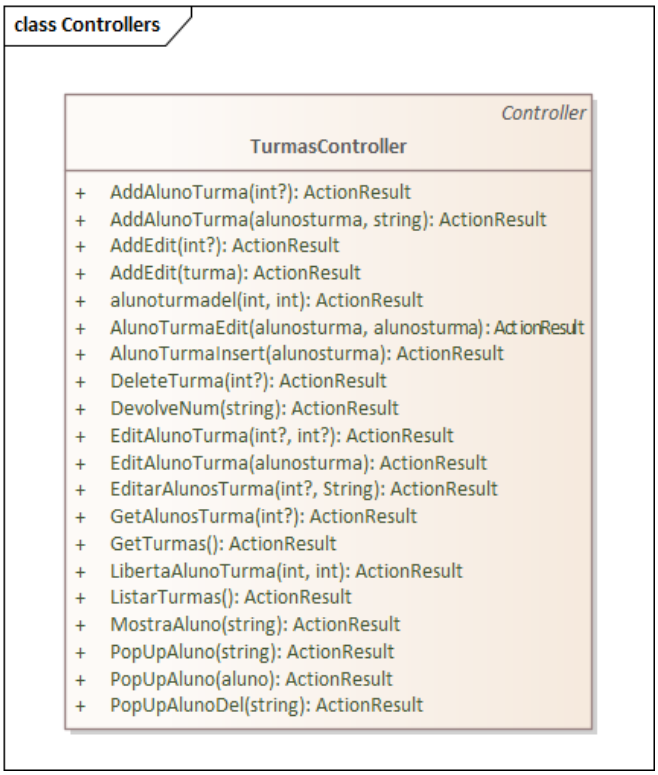
- Classe "Account Controller"

Esta classe contém a lógica associada ao controle de acessos da aplicação. Destacam-se as funcionalidades de "Login", "Registo de Contas de Utilizador", "Log off", e "Reset de Password".



- Classe "*Home Controller*"

Esta classe contém a lógica associada à gestão da entidade pessoa. A cada pessoa deverá estar associada uma de três entidades: "Docente", "Aluno e "Funcionário". A classe contempla ainda as funcionalidades associadas à inserção, edição e eliminação de registos pertencentes a cada uma destas três entidades.



- Classe "*Turmas Controller*"

Esta classe implementa as funcionalidades associadas à gestão de turmas e ainda à associação dos alunos a essas turmas. Disponibiliza ainda a edição da entidade "Aluno".

- Classe "*CursosDisciplinasController*"

Esta classe implementa as seguintes funcionalidades:

- gestão dos cursos;
- gestão das disciplinas;
- gestão dos currículos;
- gestão das aulas;
- gestão das aulas;
- gestão dos exames;
- emissão das pautas em excel.

8 LISTA DE ABREVIATURAS E SÍMBOLOS

MVC (Model View Controller) :

Corresponde a uma arquitetura de desenvolvimento de sistemas que compreende a existência de três camadas: "Model", "View" e "Controller". A "View" corresponde à camada da interface através da qual os utilizadores veiculam os seus pedidos. O "Model" representa a camada de dados em memória que correspondem às entidades do sistema. Por fim o "controller" constitui a camada onde reside a lógica da aplicação, isto é, a camada que recebe os pedidos dos utilizadores canalizados através da "View" e providencia a resposta aos mesmos tendo, eventualmente,

CRUD:

Operações de pesquisa e atualização de registos de uma base de dados (Create, Retrieve, Update, Delete)

Caso de Uso:

Segundo (Fowler,Martin. 2007) os casos de uso são uma técnica para captar os requisitos funcionais de um sistema. Eles servem para descrever as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo uma narrativa sobre como o sistema é utilizado. Em vez de descrever os casos de uso de início, acho mais fácil rodeá-los e começar descrevendo cenários. Um cenário é uma sequência de passos que descreve uma interação entre um usuário e um sistema. ...

Caso de uso extensão:

Segundo (Plíneo, Ventura 2014) Quando o caso de uso B estende o caso de uso A, significa que quando o caso de uso A for executado o caso de uso B poderá (poderá – talvez não seja) ser executado também. A direção do relacionamento é do caso de uso extensor (aqui o caso de uso B) para o caso de uso estendido (aqui o caso de uso A).

Caso de uso inclusão:

Segundo (Plíneo, Ventura 2014) quando o caso de uso A “inclui” o caso de uso B, significa que sempre que o caso de uso A for executado o caso de uso B também será executado. A direção do relacionamento é do caso de uso que está incluindo para o caso de uso incluído.

9 CAPÍTULOS

9.1 INTRODUÇÃO

A avaliação dos alunos é um componente primordial nos processos de ensino-aprendizagem pois, permite a perceção por parte dos intervenientes (docentes e discentes) dos conhecimentos e competências adquiridos nestes processos. Neste sentido, facilitar e automatizar os procedimentos relacionados com o registo de notas constitui uma economia de esforço que permite deslocar o foco da atenção naquilo que é mais importante em matéria de avaliação e que consiste na definição dos critérios e parâmetros que melhor se enquadram com a avaliação dos saberes e competências adquiridas em função das metas que, entretanto, foram preconizadas. Deste modo se depreendeu a necessidade de se agilizar e facilitar as tarefas subjacentes à elaboração, registo, homologação e consulta das pautas de avaliação.

Face ao exposto, será desenvolvida uma aplicação WEB que viria a permitir a implementação duma plataforma on-line onde todos os atores do sistema possam interagir entre si e edificar um sistema que implemente a gestão, registo e a publicação das pautas de avaliação. Neste contexto, os funcionários alocados à secretaria escolar assumem a responsabilidade do registo das seguintes entidades: pessoas, cursos, disciplinas, currículos, aulas e exames. Aos docentes é imputada a responsabilidade de registar atempadamente as avaliações decorrentes das suas provas e aos administradores a responsabilidade de homologar as pautas já preenchidas. Aos alunos é concedida a possibilidade de consultarem as pautas que são, entretanto, publicadas pelos funcionários da secretaria escolar. Os alunos têm ainda acesso ao histórico, sempre atualizado, de todas as avaliações decorrentes do seu percurso escolar. Em todos os processos da aplicação são salvaguardadas a integridade e segurança dos dados.

Face aos objetivos expostos foi feita uma reflexão acerca das tecnologias mais adequadas ao desenvolvimento do projeto. Entre as várias opções para desenvolvimento web foram ponderadas várias hipóteses tendo a escolha final recaído no ".NET FRAMWORK [2]" mais especificamente o "Asp.Net" utilizando a linguagem de programação C# e o SQL SERVER como sistema de gestão de base de dados.

A razão da escolha ter recaído nesta tecnologia "Asp.NET" (Active Server Pages) fundamentou-se essencialmente na enorme capacidade e variedade de soluções fiáveis e eficientes oferecidas por esta tecnologia e, simultaneamente, a rapidez e facilidade com que essas soluções são implementadas. A garantia oferecida pelo sistema de gestão de base de dados "SQL SERVER" no que diz respeito à integridade e segurança dos dados da aplicação constituiu também um fator importante na ponderação desta escolha.

Após o desenvolvimento da aplicação, decorreu do mesmo este relatório que procura expressar o contexto e as etapas subjacentes ao desenvolvimento dessa solução. Neste âmbito, foi esplanada a arquitetura do sistema indicando-se os respetivos requisitos funcionais e não funcionais.

Relativamente aos requisitos funcionais são enunciados e explicados os atores do sistema e quais as funções inerentes aos respetivos papéis. Ainda neste capítulo, são enunciados e caracterizados os casos de uso mais relevantes.

Seguidamente é mostrado um diagrama de classes com todas as classes que implementaram a solução indicando-se os respetivos atributos e funcionalidades.

---

[2] "*O .NET Framework é um ambiente de execução criado pela Microsoft e gerenciado para Windows que oferece uma série de serviços voltados ao desenvolvimento web, reutilizando e reaproveitando códigos, entre suas principais funções*."
Citação constante em:

" *O que é o .NET Framework? Quais as vantagens? (2022, April 11). https://enotas.com.br/blog/net-framework/*"

Após o diagrama de classes é desenhado um diagrama relacional que revela a estrutura montada no desenho da base de dados bem como as relações estabelecidas entre as várias entidades. Esta diagrama mostra a exequibilidade dos requisitos pré-definidos para a base de dados.

Após o capítulo dos requisitos funcionais é feita uma leve abordagem aos requisitos não-funcionais relevando-se, neste ponto, a importância da formação dos intervenientes do sistema.

No capítulo seguinte, "principais restrições do sistema", é referido fundamentalmente, o facto deste projeto ainda não ter sido implementado em contexto real e como tal, não ter sido exposto a todas as vicissitudes e necessidades decorrentes do dia a dia que permitiriam uma avaliação mais consubstanciada da eficiência, eficácia e potencialidade da solução.

É ainda feito um levantamento das necessidades operacionais inerentes ao projeto e ainda referidas as normas que devem nortear e regulamentar o seu funcionamento.

A finalizar o relatório são anexados os documentos com os códigos fonte correspondentes aos casos de uso mais importantes.

9.2 ARQUITETURA DO SISTEMA

9.2.1 Requisitos Funcionais

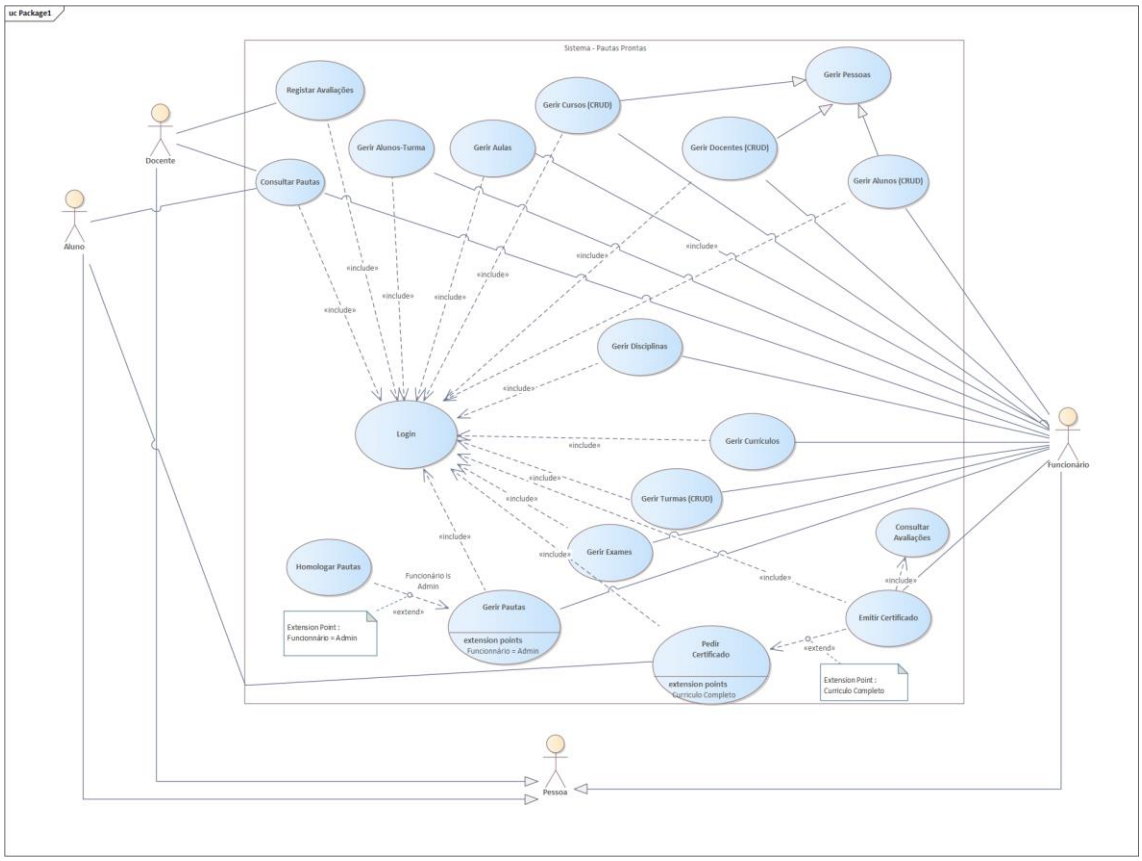


Figura 9-1 Requisitos funcionais da aplicação.

De acordo com o diagrama de "casos de uso" na figura 2, indicam-se os principais requisitos funcionais da aplicação "PautasProntas":

- Os docentes registam as avaliações nas pautas. Cada docente só tem permissão para atualizar as pautas dos exames correspondentes às aulas que lecionou;
- Todos os utilizadores autenticados poderão consultar as pautas;
- Os alunos poderão consultar o histórico das suas avaliações e no caso de terem concluído o curso poderão fazer pedido da emissão do seu certificado de habilitações;

- Os funcionários poderão emitir o certificado de habilitações sempre que o mesmo tenha sido solicitado e o aluno (sujeito do certificado) tenha obtido aprovação em todas as disciplinas do seu currículo;
- Os funcionários gerem as entidades "Cursos", "Disciplinas", "Curriculos", "Aulas", "Exames" e "Pautas". Esta gestão implica a capacidade de executar as operações CRUD zx[3]nestas entidades ;
- Se o funcionário estiver associado a um conta de grupo de administrador poderá homologar as pautas;
- Os funcionários afetos aos serviços da secretaria escolar poderão imprimir as pautas em excel;

[3] CRUD - Abreviatura dos termos Create, Retrieve, Update e Delete. Este termo é empregue para definir as operações de pesquisa e atualização dos registos de uma tabela de uma base de dados

9.2.1.1 dentificação dos Atores do Sistema

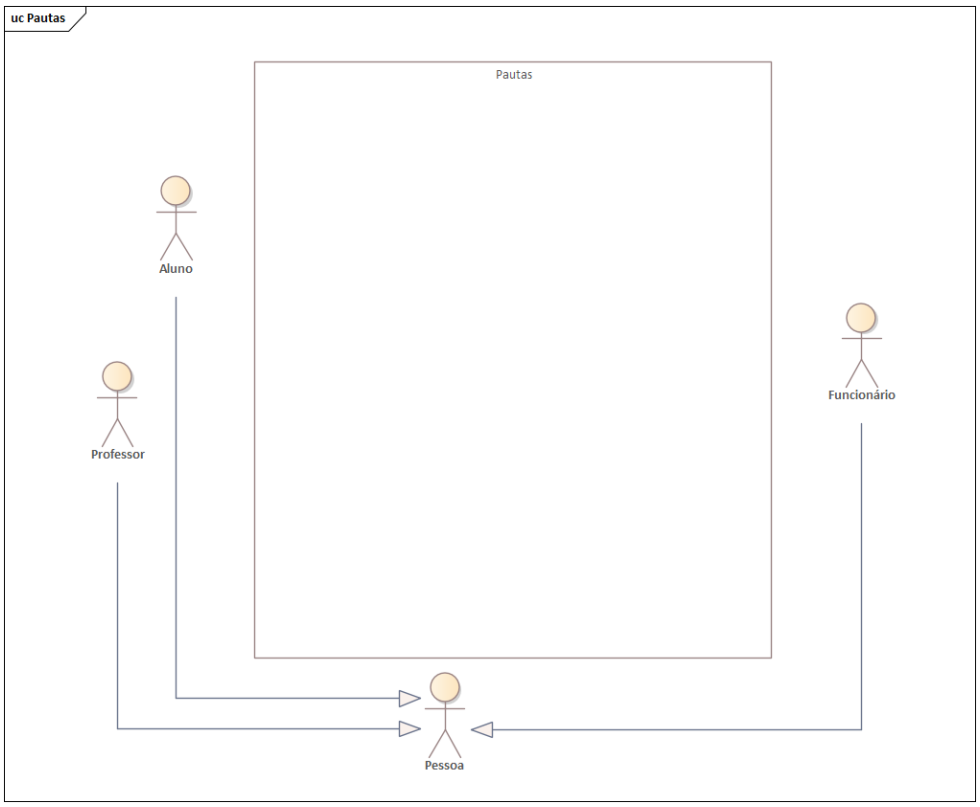


Figura 9-2 - Atores integrados no sistema

De acordo com a figura 2-2 descrevem-se de seguida, os atores incluídos no sistema.

9.2.1.1.1 <Aluno>

Todo o elemento da instituição que está matriculado e frequenta ou que já frequentou a instituição no papel de aluno. O sistema permitirá que o aluno consulte as pautas da instituição e o histórico das suas avaliações.

O Aluno poderá solicitar através da aplicação a emissão do seu certificado de habilitações.

9.2.1.1.2 <Professor>

Elemento que leciona na instituição. O professor leciona um ou vários currículos correspondendo cada currículo a uma disciplina ministrado no contexto de um curso. Ministrará uma ou várias aulas sendo que, cada aula está associada a um currículo e a uma turma. É da sua responsabilidade o preenchimento das pautas com as avaliações decorrentes dos exames correspondentes às suas aulas.

9.2.1.1.3 <Funcionário>

O funcionário é o elemento que age essencialmente como ator secundário dando apoio às respostas do sistema nas seguintes tarefas:

- Gestão das entidades: alunos, docente e funcionários;
- Gestão dos cursos disciplinas e currículos;
- Gestão das aulas;
- Gestão dos exames;
- Emissão de certificados.

9.2.1.1.4 <Pessoa>

A entidade "Pessoa" generaliza os três atores anteriores (professor, docente e aluno).

9.2.1.2 Principais Casos de Utilização

9.2.1.2.1 <Gerir Entidades: Aluno, Professor, Funcionário e Pessoa>

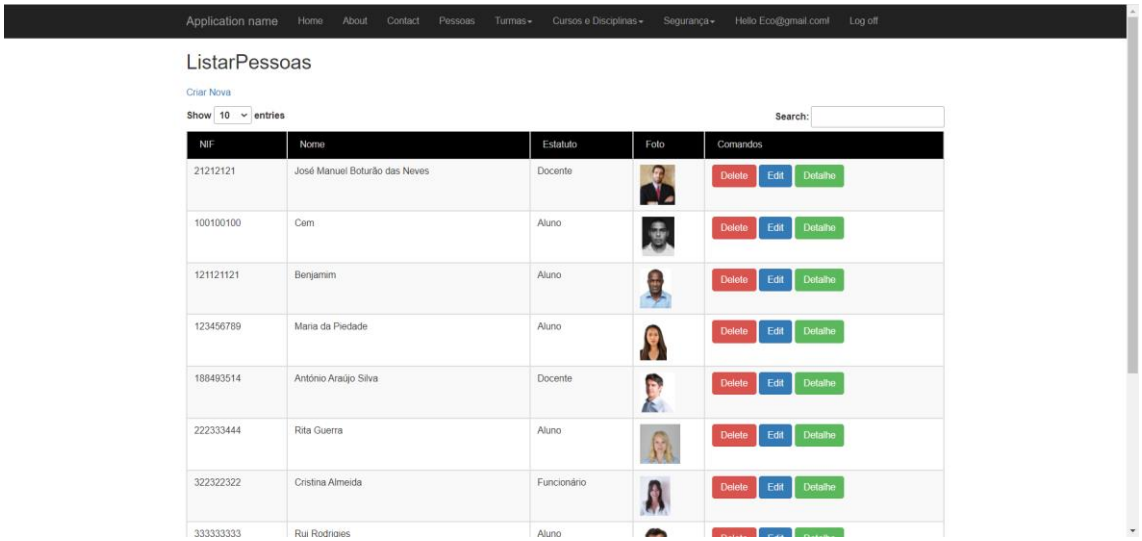


Figura 9-3 - View [4]Listar Pessoas

4 View : Razor Page : Página web composta por HTML, estilos e C#. Cada razor page está associada a um método no controller o qual corresponde a uma determinada rota a qual poder ser entendida como um pedido.

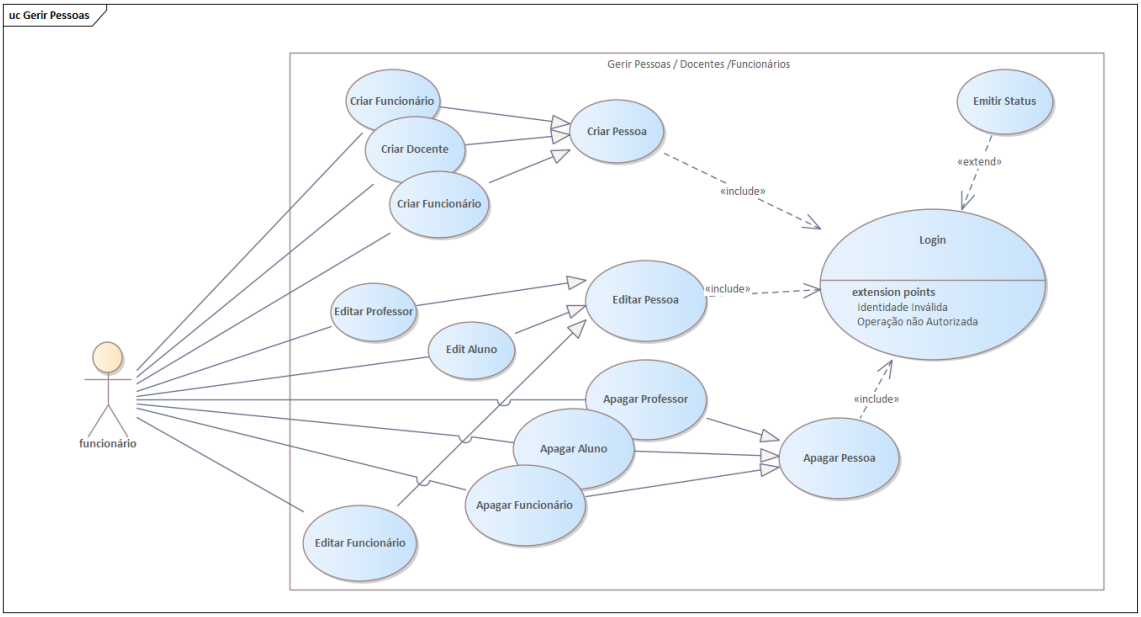


Figura 9-4 – Caso de uso – Gerir Entidades

9.2.1.2.1.1 Descrição do Use case:

Identificação do Caso de Uso	1
Nome do Caso de Uso	**Gerir Pessoas (Professor, Aluno, Funcionário)**
Criado. Por:	José Neves
Data:	Setembro 2021
Descrição:	O funcionário, desde que autenticado e autorizado, poderá efetuar as operações CRUD nas entidades Professor, Aluno e Funcionário. Estas três entidades derivam da entidade Pessoa.
Casos de uso inclusão:	Login
Casos de uso extensão	Emissão de Status: Autenticado e Autorizado
Atores Primários:	Funcionário
Atores Secundários:	
Pré-Condições:	O funcionário deve estar autenticado e as entidades sujeitas às operações de edição e eliminação não poderão ter quaisquer dependências.
Pós-Condições:	

Fluxo Principal:	1. Funcionário Faz Login 2. Opta por realizar a operação de atualização. 3. O registo é eliminado, editado ou inserido 4. O sistema devolve um "feedback" descrevendo o sucesso da operação
Fluxo Alternativo	1b) O funcionário não está autenticado ou autorizado para realizar a operação. 1b.1) Mensagem do sistema indicando o status do utilizador; 1b.2) Fim do "use case"
	2b) Se o registo tem dependência a atualização é inviabilizada. 2b.1) Sistema devolve mensagem com o motivo do insucesso da operação

9.2.1.2.1.2 Diagrama de Sequência – Gerir Entidades:

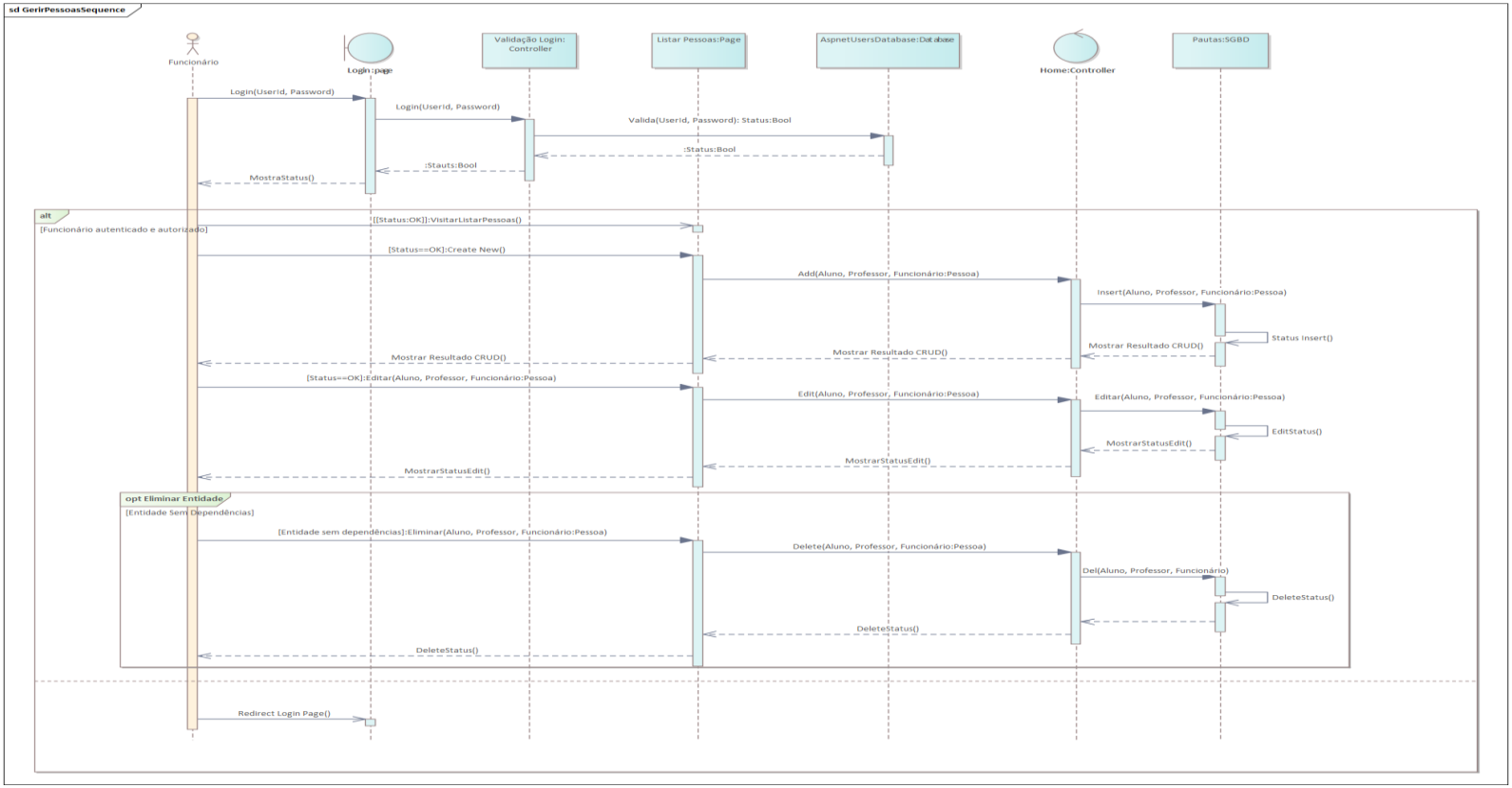


Figura 9-5 - Diagrama sequência - Gerir Entidades

9.2.1.2.1.3 Diagrama de Classe – "Home Conroller":

Figura 9-6 Diagrama de Classe "Home Controller"

9.2.1.2.1.4 A implementação deste use case está incluída no Anexo B.

9.2.1.2.2 <Gerir Turmas>

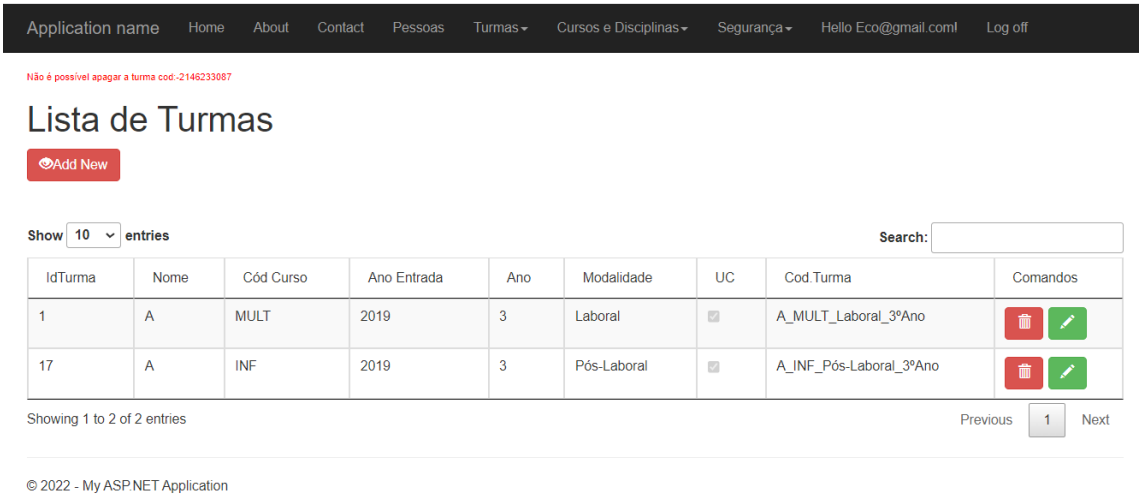


Figura 9-7 View - Listar Turmas

9.2.1.2.2.1 - <Use Case Diagram – Gerir Turmas>

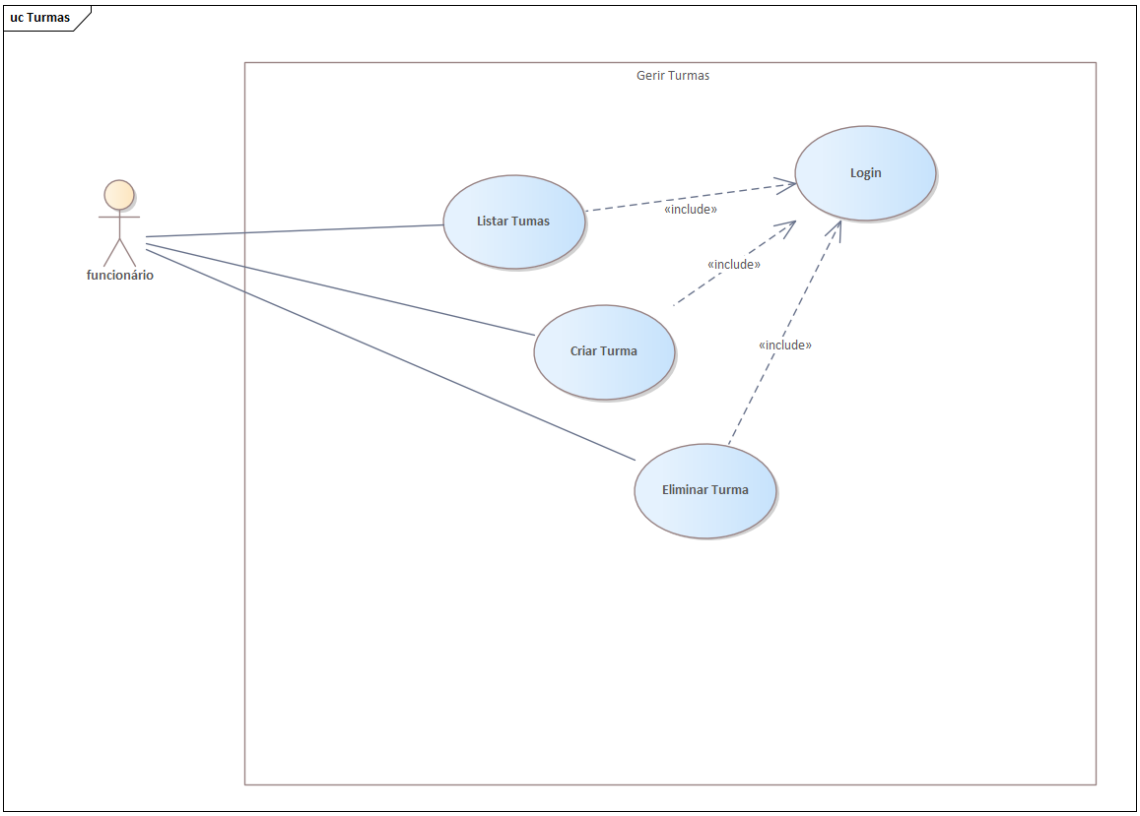


Figura 9-8 Diagrama de caso de uso <Gerir Turmas>

9.2.1.2.2.2 - <Descrição do caso de uso– Gerir Turmas>

Identificação do Caso de Uso	2
Nome do Caso de Uso:	Gerir Turmas
Criado Por:	José Neves
Data:	Novembro 2021

Descrição:	O funcionário após login poderá aceder à página “ListarTurmas”. Nesta página poderá optar pelas seguintes opções: criar nova turma, editar turma e eliminar turma.
Caso de Uso Inclusão:	Login
Casos de uso extensão	
Atores Primários:	Funcionário
Atores Secundários:	
Pré-Condições:	Para fazer as operações de atualização o funcionário terá de estar autenticado e autorizado. Para a operação "eliminar turma" será necessário que a mesma não tenha dependências isto é: alunos ou aulas associadas.
Pós-Condições:	
Fluxo Principal:	1- Funcionário autentica-se
	2 - Funcionário acede à página "Listar turmas"
	3 -Acede à uma das opções de atualização:

	Insert, Update ou Delete
	4 - O sistema executa e devolve o estado da operação anterior
Fluxo Alternativo	1b - Falha de Login
	1b1 – Utilizador é redirecionado para a página de Login e o sistema e o caso de uso retorna à etapa 2.
	3b- A turma tem dependências (alunos/aulas)
	3b1-Fim do Caso de Uso.

9.2.1.2.2.3 - <Diagrama de Sequência – Gerir Turmas>

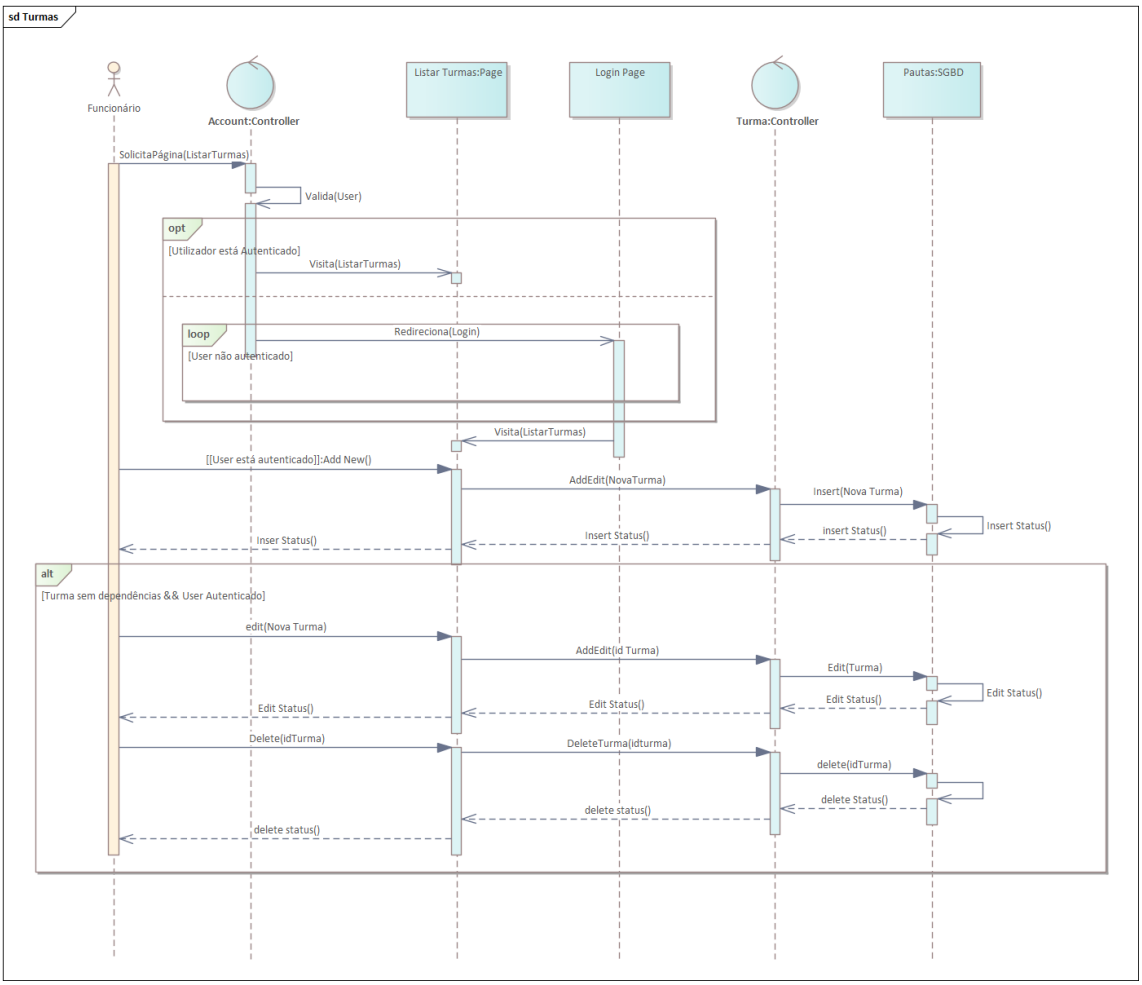


Figura 9-9 Diagrama de sequência <Gerir Turmas>

9.2.1.2.2.4 - <Diagrama de Classe : TurmaController>

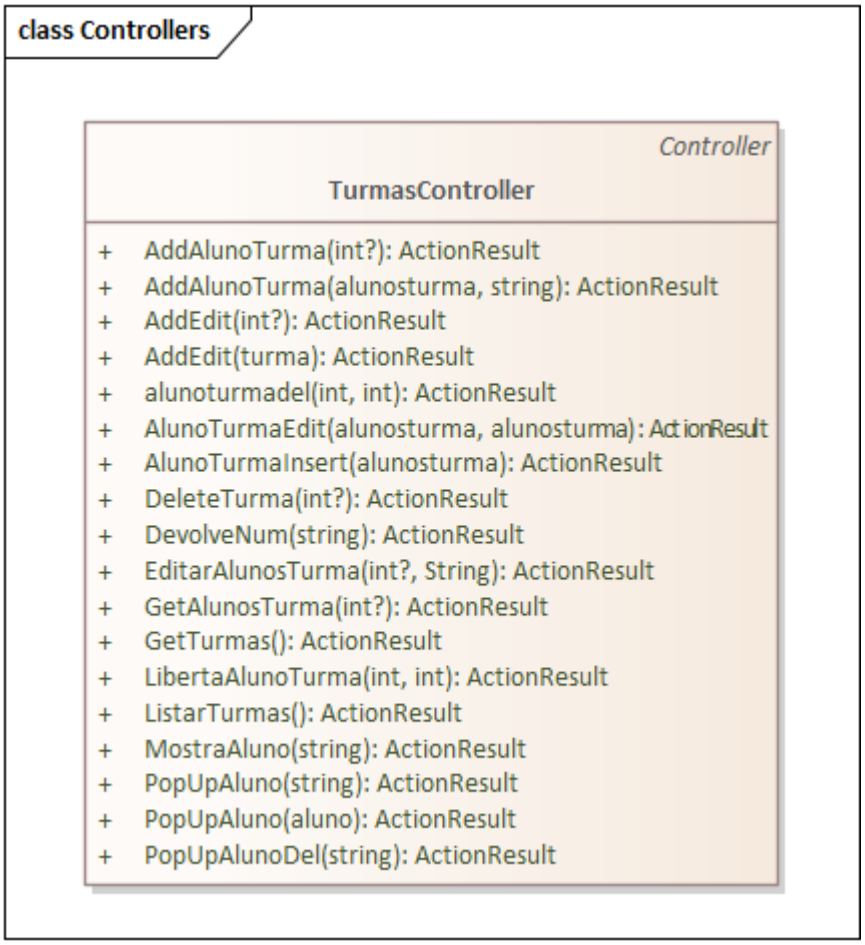


Figura 9-10 - Diagrama de classe <Turma Controller>

9.2.1.2.2.5 Modelo Relacional da Entidade Turma

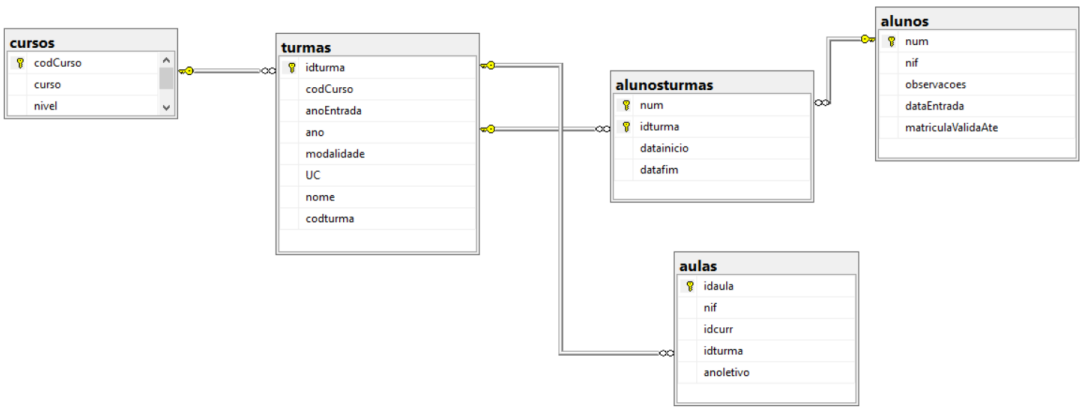


Figura 9-11 modelo relacional associado à entidade Turma

9.2.1.2.2.6 Implementação do caso de uso <Anexo C Turmas Controller>

9.2.1.2.3 <Gerir Alunos Turma>

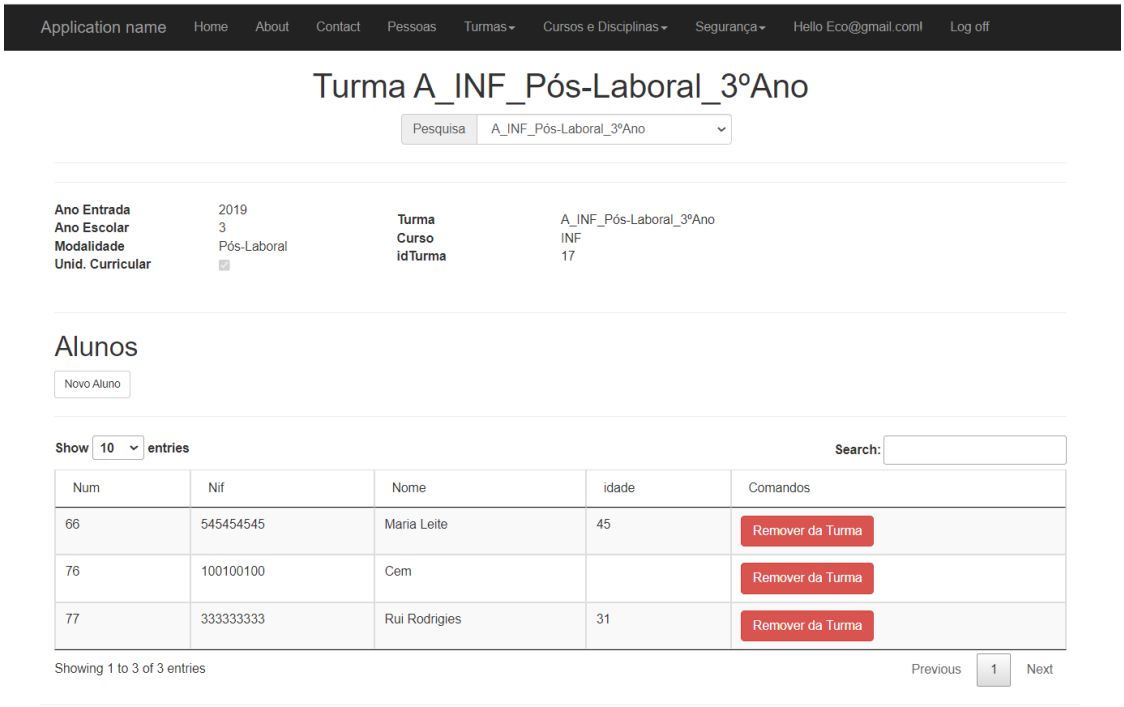


Figura 9-12 View <Gerir alunos por Turma>

9.2.1.2.3.1 - <Diagrama de caso de uso – Gerir Alunos Turma>

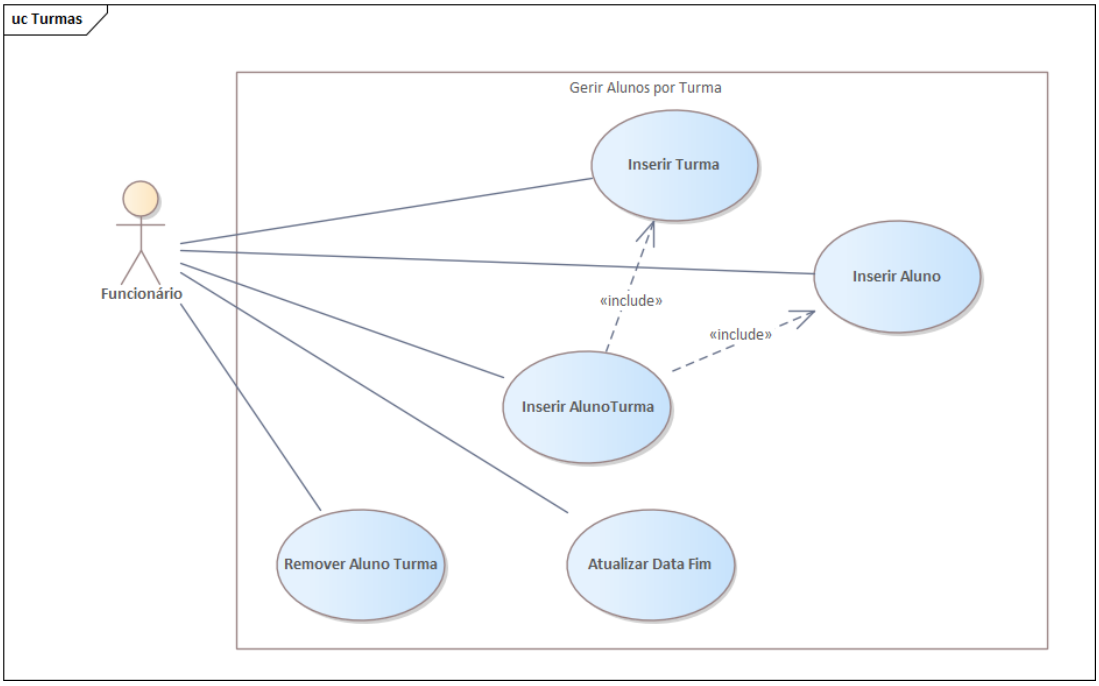


Figura 9-13 Diagrama de caso de uso <Gerir Alunos Turma>

9.2.1.2.3.2 Descrição do Caso de Uso

Identificação do Caso de Uso:	3
Nome do Caso de Uso	Gerir alunos turma
Criado Por:	José Neves
Data:	Novembro 2021
Descrição:	O funcionário implementa as operações CRUD na entidade "*alunosTurma*". O aluno termina o seu vínculo à turma com o preenchimento do campo datafim.
Caso de Uso Incluídos:	InserirTurma, InserirAluno

Casos de uso extensão	
Atores Primários:	Funcionário
Atores Secundários:	
Pré-Condições:	Para associar um aluno à turma é necessária a pré-existência da turma e do aluno.
Pós-Condições:	A entidade não pode ser eliminada ou editada se tiver registos dependentes da entidade "ExameNotas",
Fluxo Principal	1 - Funcionário insere uma nova entidade AlunoTurma; 2 - O Funcionário edita a entidade aluno turma podendo alterar os campos "num", "idturma", "datainicio" e "datafim"; 3- O Funcionário elimina a entidade "AlunoTurma";
Fluxo Alternativo	2b- A entidade "alunoTurma" não poderá ser alterada se existirem dependências na entidade "Exame Notas"; 2b.1 - Fim do use case; 3b- A entidade "alunoTurma" não poderá ser eliminada se existirem dependências na entidade "Exame Notas"; 3b.1 - Fim do "Use Case";

9.2.1.2.3.3 Modelo relacional associado à entidade "alunosTurmas".

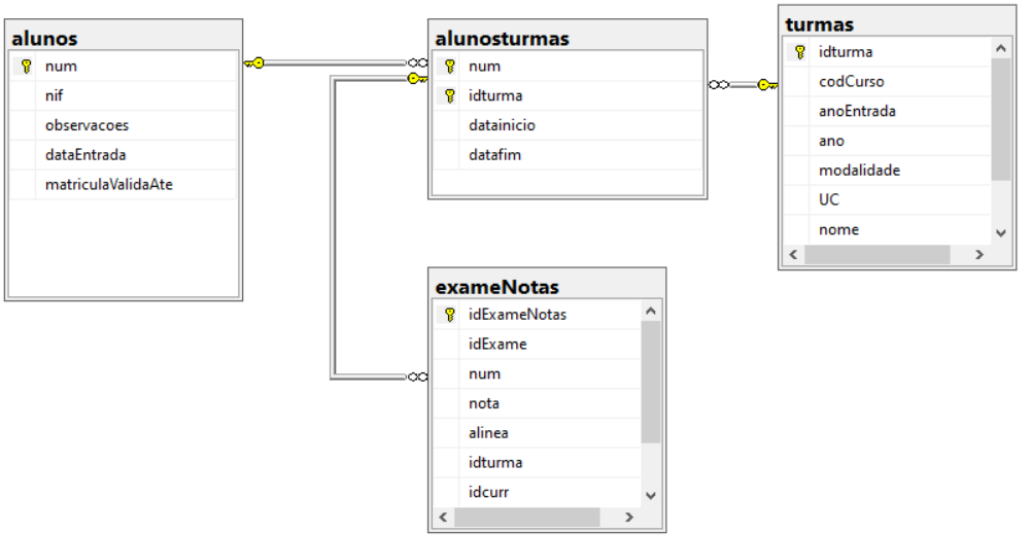


Figura 9-14 - Modelo Relacional associado à entidade "alunosTurmas".

9.2.1.2.3.4 Implementação do caso de uso Gerir Alunos Turma -Anexo C

9.2.1.2.4 *<Caso de uso Gerir Cursos>*

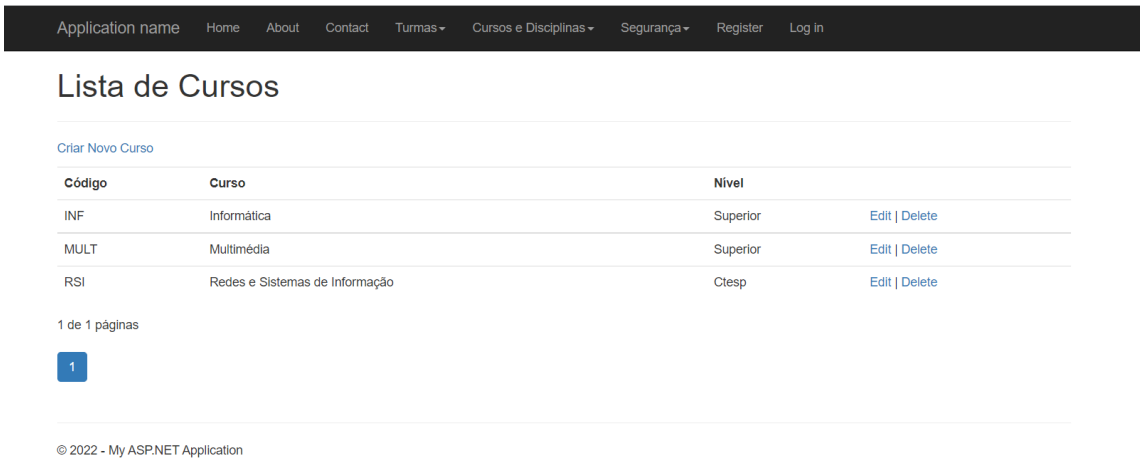
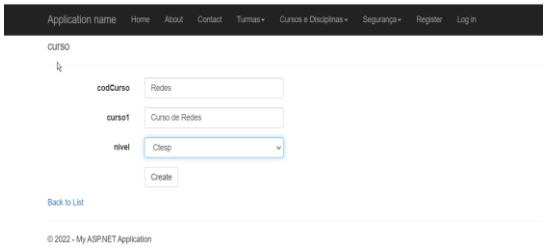


Figura 9-15 View "Listagem de cursos"

O objetivo das funcionalidades associadas a este "caso de uso" relacionam-se com a gestão dos cursos ministrados na instituição e inclui das seguintes funcionalidades:

- Inserir novo curso:



- Editar novo Curso:

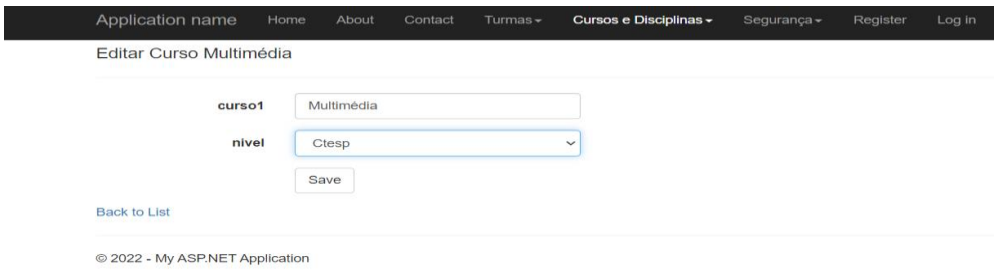


Figura 9-16 View "Editar Curso"

- Eliminar curso.

9.2.1.2.4.1 Diagrama "Relacional" associado à entidade "Curso".

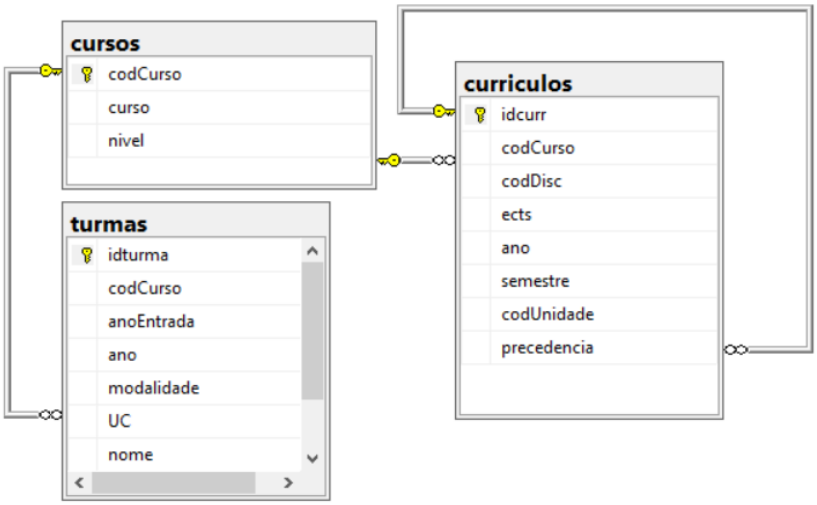


Figura 9-17 - Modelo relacional associado à entidade "Curso"

9.2.1.2.4.2 Diagrama Use case

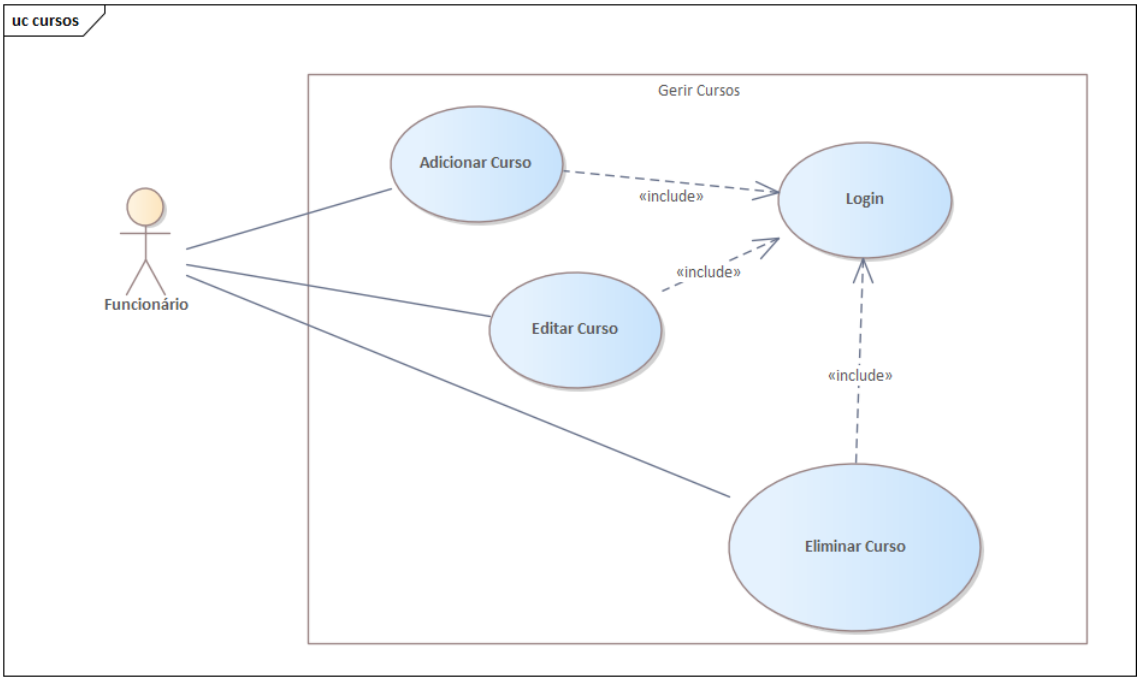


Figura 9-18 - diagrama do caso de uso <Gerir Cursos>

9.2.1.2.4.3 Descrição caso de usos "Gerir Cursos"

Identificação do Caso de Uso	4
Nome do Caso de Uso	Gerir Cursos
Criado Por:	José Neves
Data:	Novembro de 2021

Descrição:	Este "Use Case" inclui as funcionalidades: a) Inserir curso; b) Editar Curso; c) Eliminar Curso
Casos de uso Inclusão:	Login
Casos de uso extensão	
Atores Primários:	Funcionário
Atores Secundários:	
Pré- Condições:	O user deverá estar autenticado. Para proceder a alterações na entidade é necessário que não haja dependências associadas às entidades "turma" e "currículo"
Pós-Condições:	
Fluxo Principal	1A) O funcionário solicita operação de Procurar, Inserir, editar e Eliminar; 2) O sistema executa a operação CRUD na Base de Dados; 3) O sistem devolve o estado da operação realizada;

Fluxo Alternativo	1B) O utilizador não está autenticado e o sistema remete-o para a página de Login; 1B.1) O utlizador autentica-se e o sistema volta ao passo 1A 1B.2) As credencias do funcionário estão erradas e o use-case termina; 2B) O registo tem dependências e as operações CRUD são inviabilizadas; 2B.1) Sistema devolve status da operação; 2B.2) Fim do Use-Case;

9.2.1.2.4.4 Diagrama de Sequência "Gerir Curso"

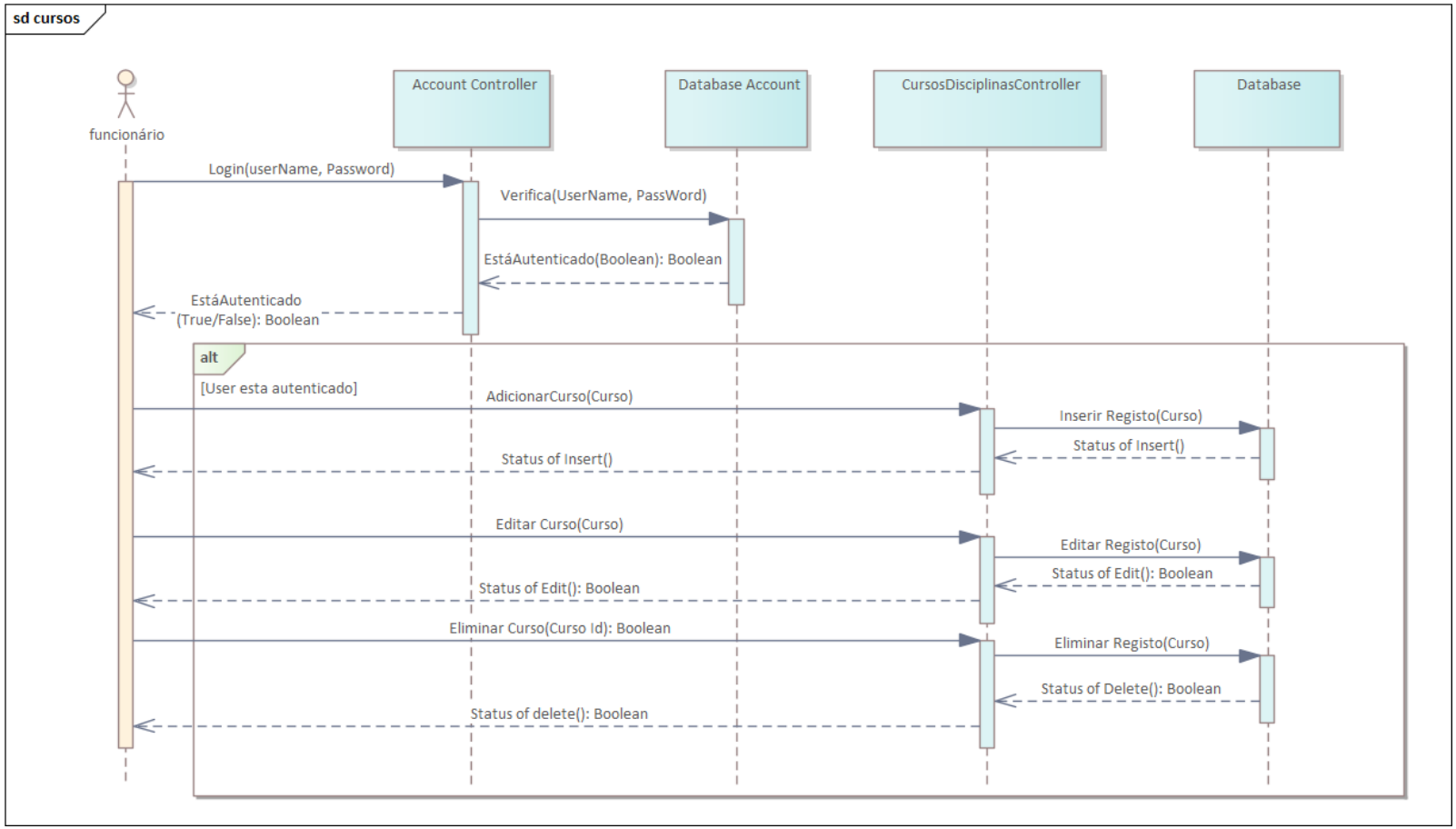


Figura 9-19 - Diagrama de sequência do caso de uso Gerir Cursos

9.2.1.2.4.5 <Gerir Disciplinas>

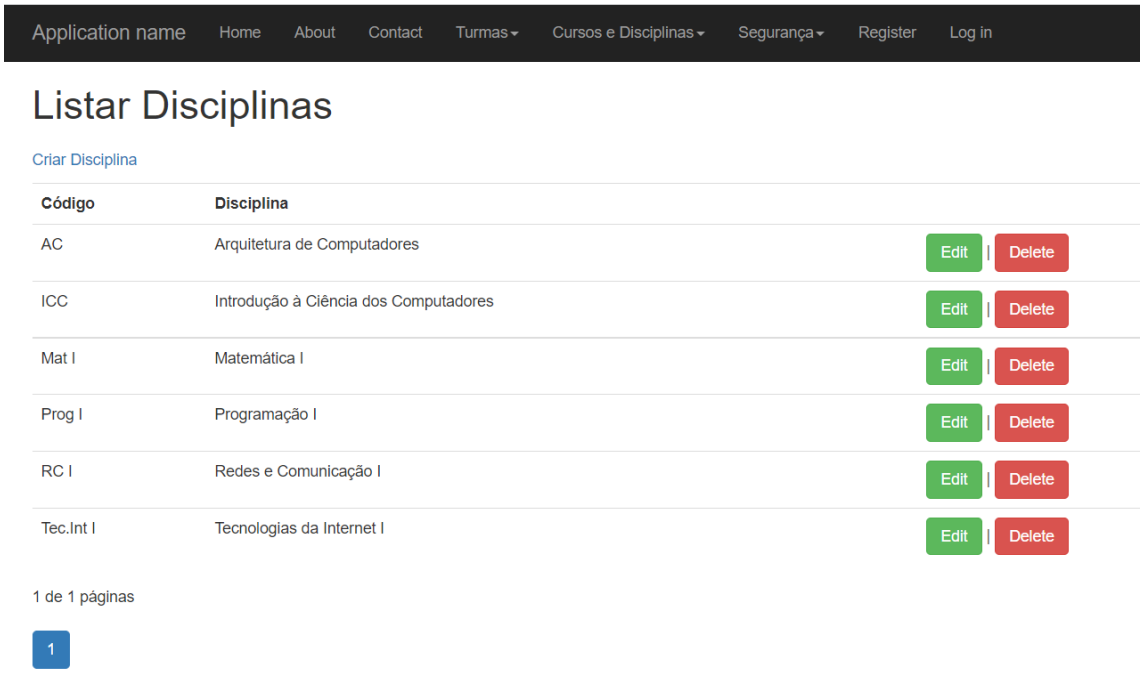


Figura 9-20 - View Listar Disciplinas

As funcionalidades destes caso-de-uso relacionam-se com as operações CRUD à entidade disciplina.

9.2.1.2.4.5.1 <modelo relacional associado à entidade disciplina>

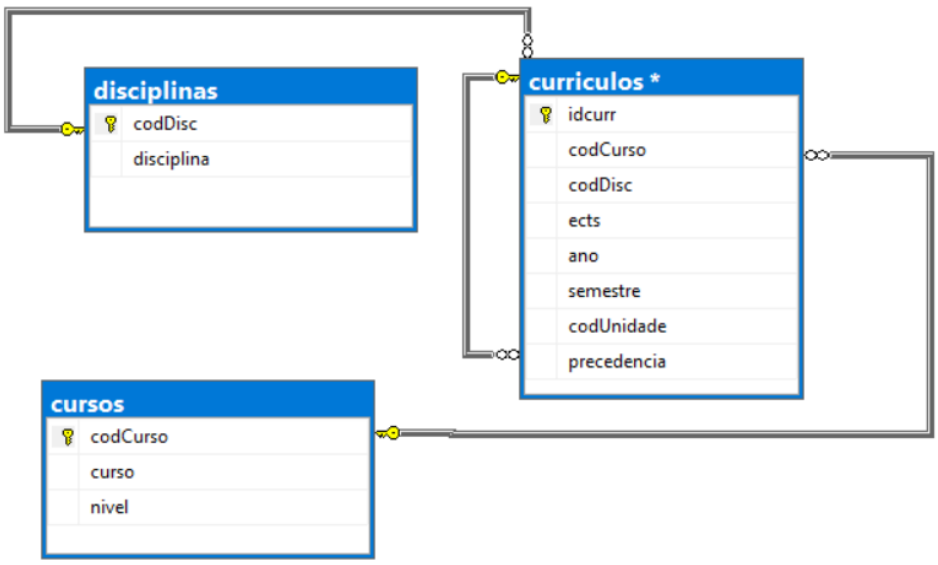


Figura 9-21 - Modelo Relacional associado à entidade disciplina

De acordo com o diagrama anterior uma disciplina pode estar associada a mais do que um curso e vice-versa, isto é, um curso está associado a várias disciplinas;

Esta relação é implementada pelo ternário composto pelas entidades "Currículo" e as outras duas entidades já mencionadas. Saliente-se ainda, a relação reflexiva da entidade currículo que permite incorporar o conceito de precedência entre disciplinas.

9.2.1.2.4.5.2 <Diagrama Caso-de-Uso da entidade "Disciplina">

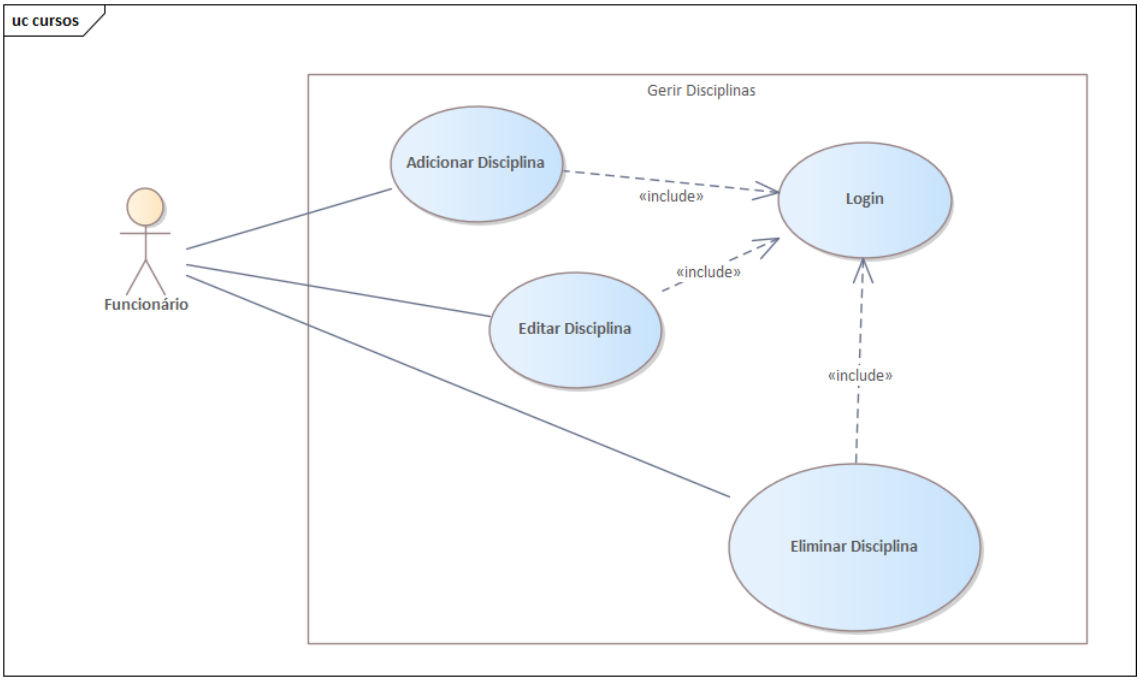


Figura 9-22 - diagrama de caso de uso <gerir disciplinas>

O funcionário após autenticar-se poderá listar, inserir, editar e eliminar disciplinas;

9.2.1.2.4.5.3 <Descrição do caso de uso "Gerir Disciplinas">

Identificação do Caso de Uso	5
Nome do Caso de Uso	Gerir Disciplinas
Criado Por:	José Neves
Data:	Novembro de 2021

Descrição:	Este "Use Case" inclui as funcionalidades: a) Inserir disciplina; b) Editar disciplina; c) Eliminar disciplina
Caso de Uso Incluídos:	Login
Casos de uso extensão	
Atores Primários:	Funcionário
Atores Secundários:	
Pré-Condições:	O utilizador deverá estar autenticado. Para proceder a alterações na entidade é necessário que não haja dependências associadas à entidade "currículo"
Pós-Condições:	
Fluxo Principal	1A) O funcionário solicita operação de Procurar, Inserir, editar e Eliminar; 2) O sistema executa a operação CRUD na Base de Dados; 3) O sistema devolve o estado da operação realizada;
Fluxo Alternativo	1B) O utilizador não está autenticado e o sistema remete-o para a página de Login; 1B.1) O utilizador autentica-se e o sistema volta ao passo 1A 1B.2) As credencias do funcionário estão erradas e o use-case termina; 2B) O registo tem dependências e as operações CRUD são inviabilizadas; 2B.1) Sistema devolve status da operação; 2B.2) Fim do Use-Case;

9.2.1.2.4.5.4 <Diagrama de sequência do caso de uso “Gerir disciplinas”>

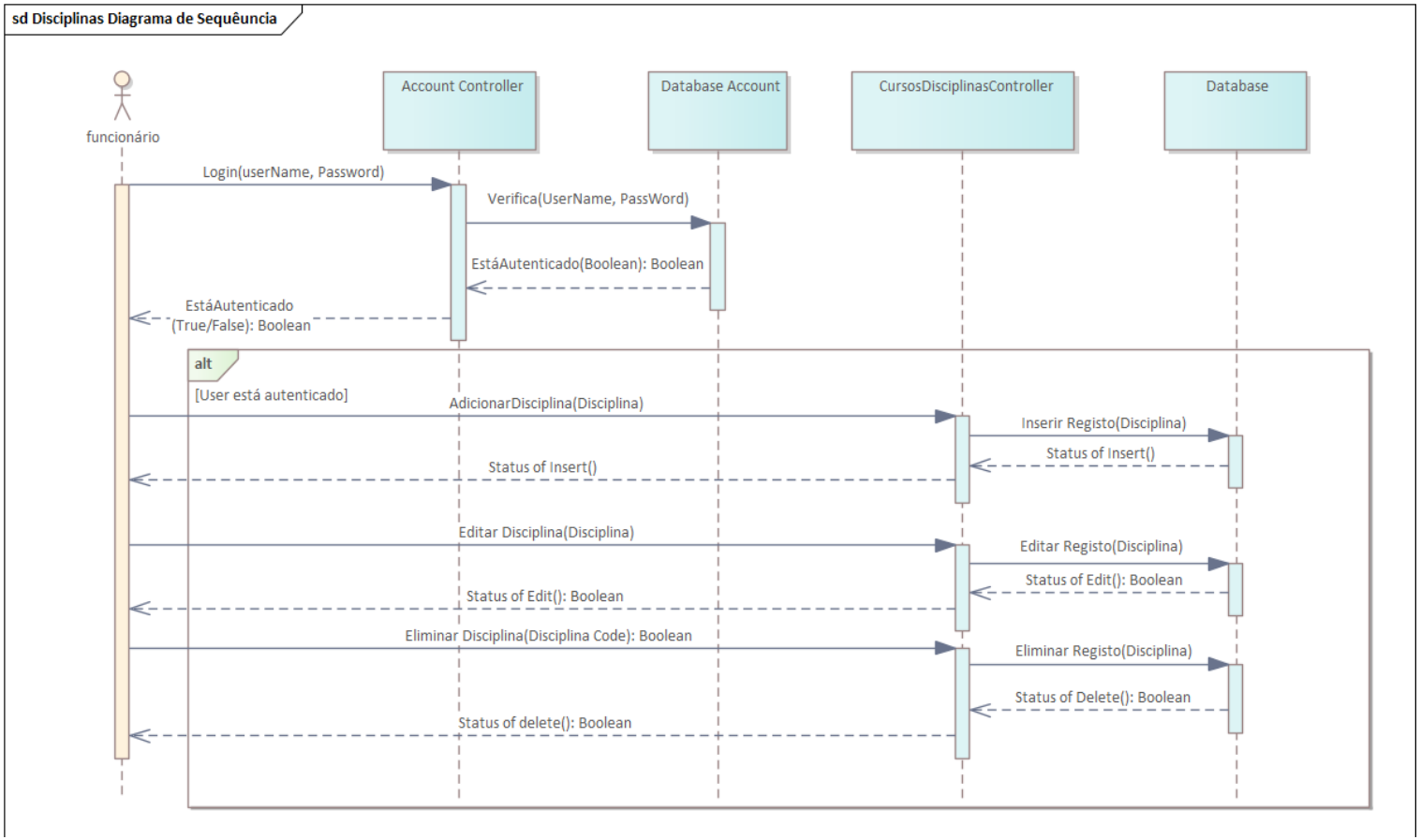


Figura 9-23 - Diagrama de sequência <Gerir disciplinas>

9.2.1.2.4.6 Implementação do caso de uso gerir disciplinas – Anexo D <Cursos Disciplinas Controller>

9.2.1.2.4.7 Implementação na lógica do cliente do caso de uso gerir disciplinas – Anexo I - <VIEW Listar Disciplinas>

9.2.1.2.4.8 <Gerir Currículos>

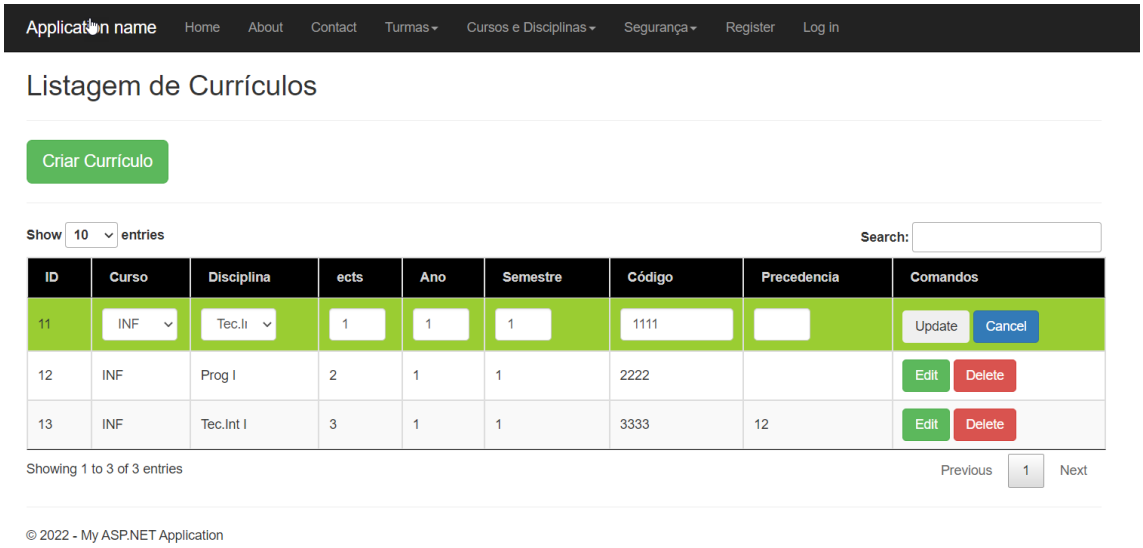


Figura 9-24 - View Listagem de currículos

Este caso de uso implementa as seguintes funcionalidades:

- Listar currículos;
- Pesquisar currículos;
- Paginar Currículos;
- Inserir Currículo;
- Editar Currículo;
- Eliminar currículo;

Para a sua realização foi implementada uma "datatable" (componente de jquery) e as funcionalidades CRUD foram implementadas através de chamadas assíncronas recorrendo-se a AJAX.

9.2.1.2.4.8.1 <Modelo Relacional associado à entidade Currículo>

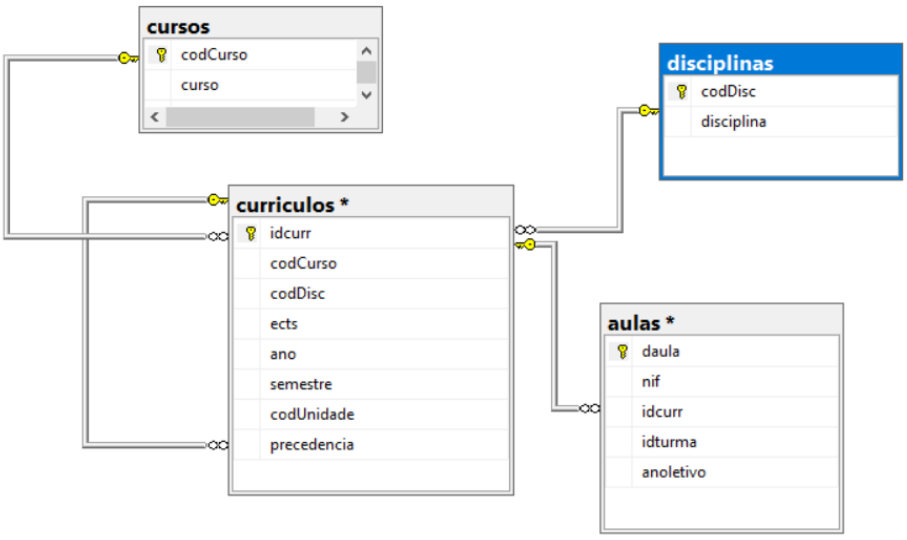


Figura 9-25 - Modelo relacional associado à entidade currículo

De acordo com o esquema anterior depreende-se que a "entidade-associação" currículo permite uma associação de muitos-para-muitos entre as entidades Disciplina e Curso.

Percebe-se igualmente que um currículo está associado a várias aulas.

9.2.1.2.4.8.2 <Diagrama do caso-de-uso associado à entidade currículo>

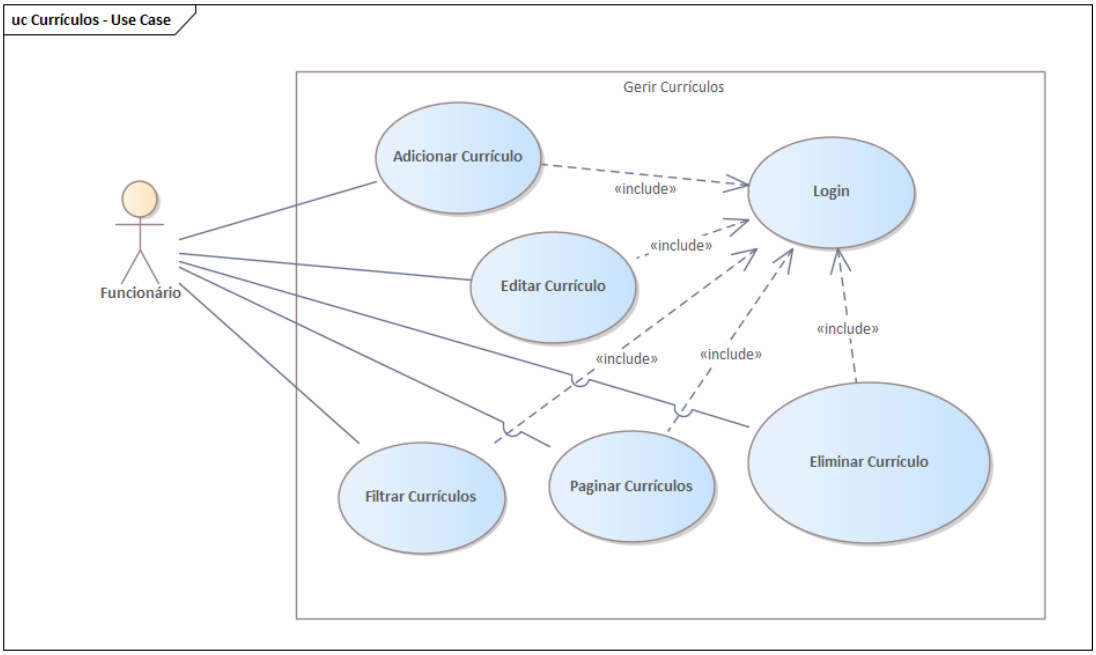


Figura 9-26 - Diagrama do caso de uso <Gerir Currículos>

9.2.1.2.4.8.3 <Descrição do caso de uso – Gerir Currículos>

Identificação do Caso de Uso	6
Nome do Caso de Uso	Gerir Currículos
Criado Por:	José Neves
Data:	Novembro de 2021

Descrição:	Este "Use Case" inclui as funcionalidades: a) Inserir Currículo; b) Editar Currículo; c) Eliminar Currículo; d) Filtrar/Pesquisar Currículos; e) Paginar Currículos;
Caso de Uso Inclusão:	Login
Casos de uso extensão	
Atores Primários:	Funcionário

Atores Secundários:	
Pré-Condições:	O utilizador deverá estar autenticado. Para proceder a alterações na entidade é necessário que não haja dependências associadas à entidade "Aulas"
Pós-Condições:	
Fluxo Principal:	1A) O funcionário solicita operação de Procurar, Inserir, editar e Eliminar; 2) O sistema executa a operação CRUD na Base de Dados; 3) O sistema devolve o estado da operação realizada;
Fluxo Alternativo	1B) O utilizador não está autenticado e o sistema remete-o para a página de Login; 1B.1) O utilizador autentica-se e o sistema volta ao passo 1A 1B.2) As credencias do funcionário estão erradas e o use-case termina; 2B) O registo tem dependências e as operações CRUD são inviabilizadas; 2B.1) Sistema devolve status da operação; 2B.2) Fim do Use-Case;

9.2.1.2.4.8.4 Diagrama de sequência – Gerir Currículos

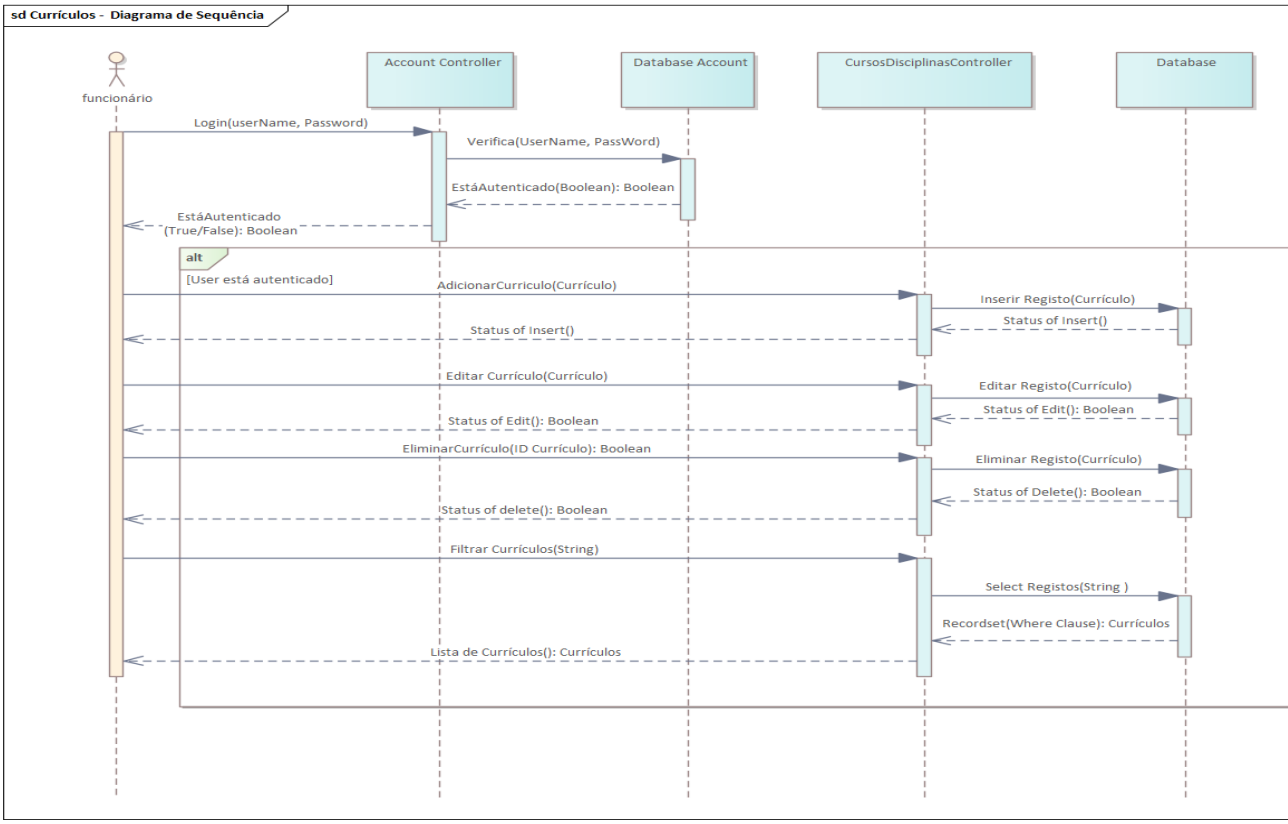


Figura 9-27 Diagrama de sequência do caso de uso <Gerir currículos>

9.2.1.2.4.8.5 Implementação do caso de uso gerir currículos – <Anexo D - Controller>

9.2.1.2.4.8.6 Implementação do caso de uso Gerir Currículos na lógica do cliente – <View Currículos>

9.2.1.2.4.9 < Caso de uso Gerir Aulas>

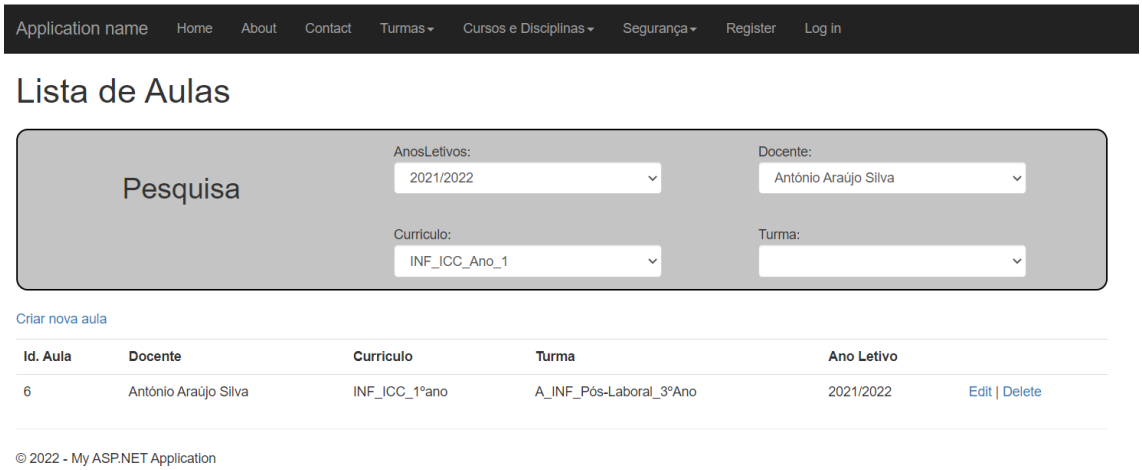


Figura 9-28 – View <Lista de Aulas>

9.2.1.2.4.9.1 Diagrama relacional relacionado com a entidade "aula".

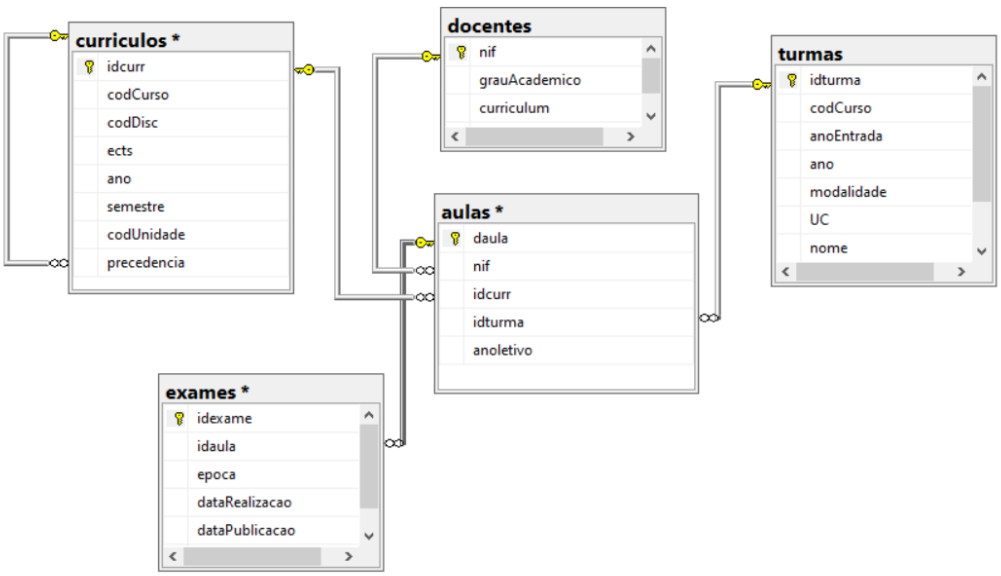


Figura 9-29 - Diagrama relacional associado à entidade <Aula>

O conceito de aula nasce da associação entre um docente, um currículo e uma turma; no contexto de um determinado ano letivo. Uma aula consite numa associação ternária entre um currículo (disciplina/turma), um docente e uma turma.

Neste caso de uso são implementadas as seguintes funcionalidades relacionadas com a entidade “Aula”:

- ✓ Filtrar e Pesquisar aulas;
- ✓ Inserir aula;
- ✓ Editar aula;
- ✓ Eliminar aula.

A cada aula estão associadas várias instâncias da entidade exames.

9.2.1.2.4.9.2 Diagrama de caso de uso da gestão das aulas

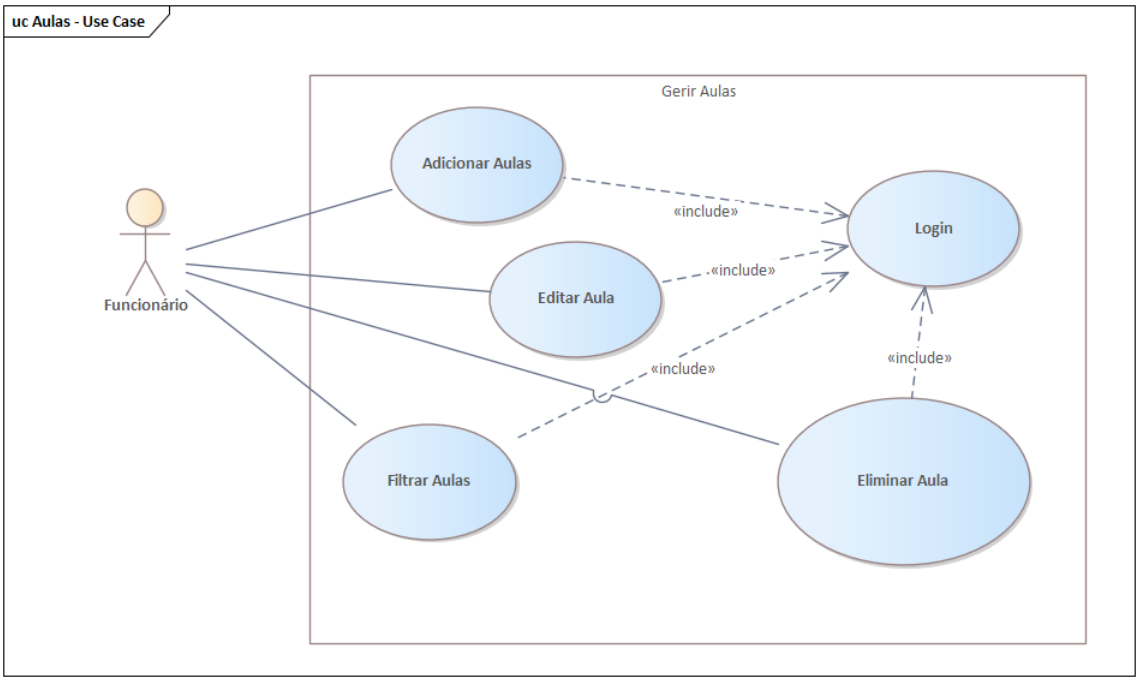


Figura 9-30 - diagrama de caso de uso - Gerir aulas

9.2.1.2.4.9.3 Diagrama de sequência do caso de uso “Gerir Aulas”

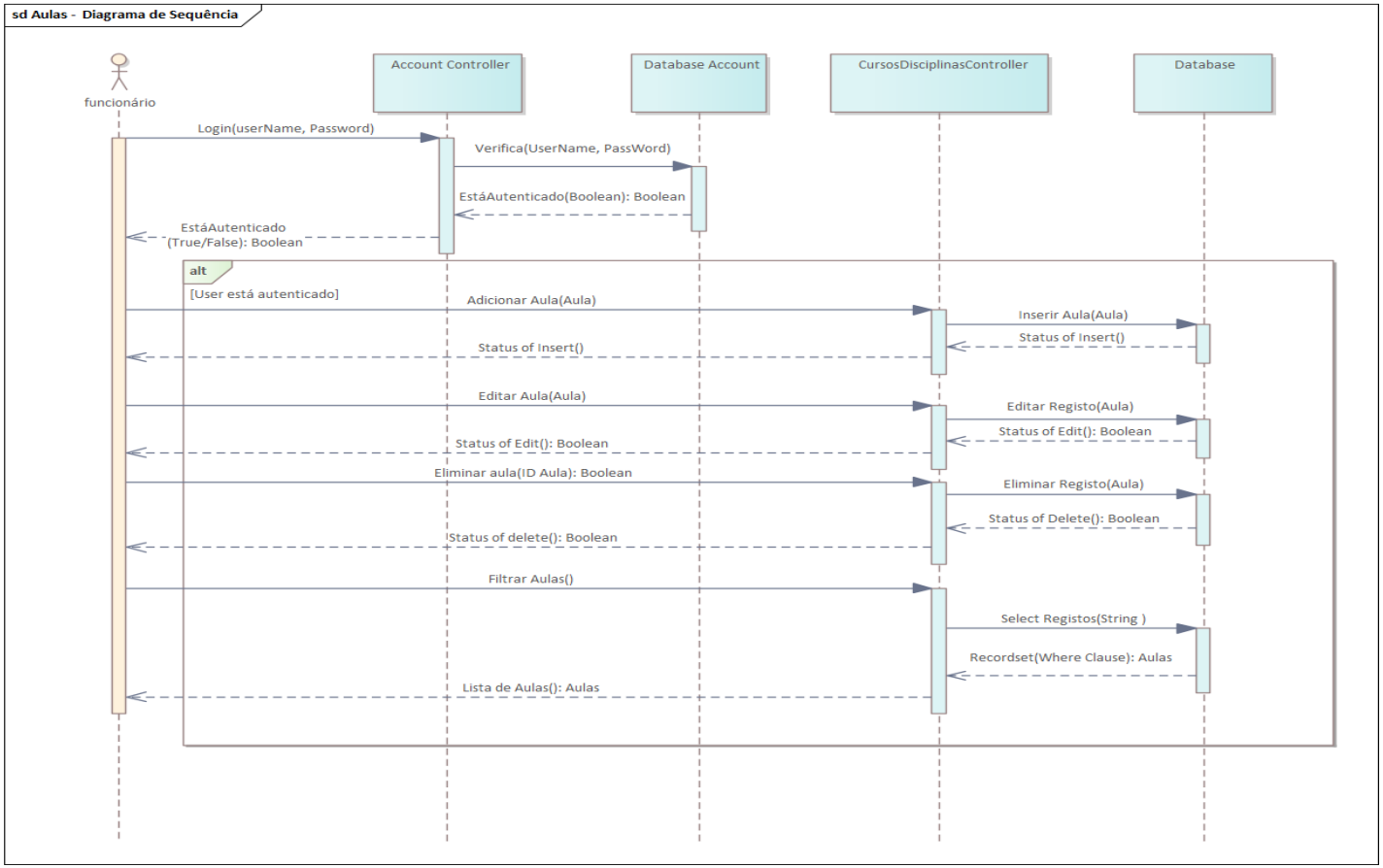


Figura 9-31 diagrama de sequência associado ao caso de uso <Gerir Aulas>

9.2.1.2.4.9.4 Implementação do caso de uso <Gerir Aulas> - Anexo D

9.2.1.2.4.9.5 Implementação na lógica do cliente do caso de uso ANEXO H <View Listar Aulas>

-

9.2.1.2.4.10 <Gerir Exames e Pautas>

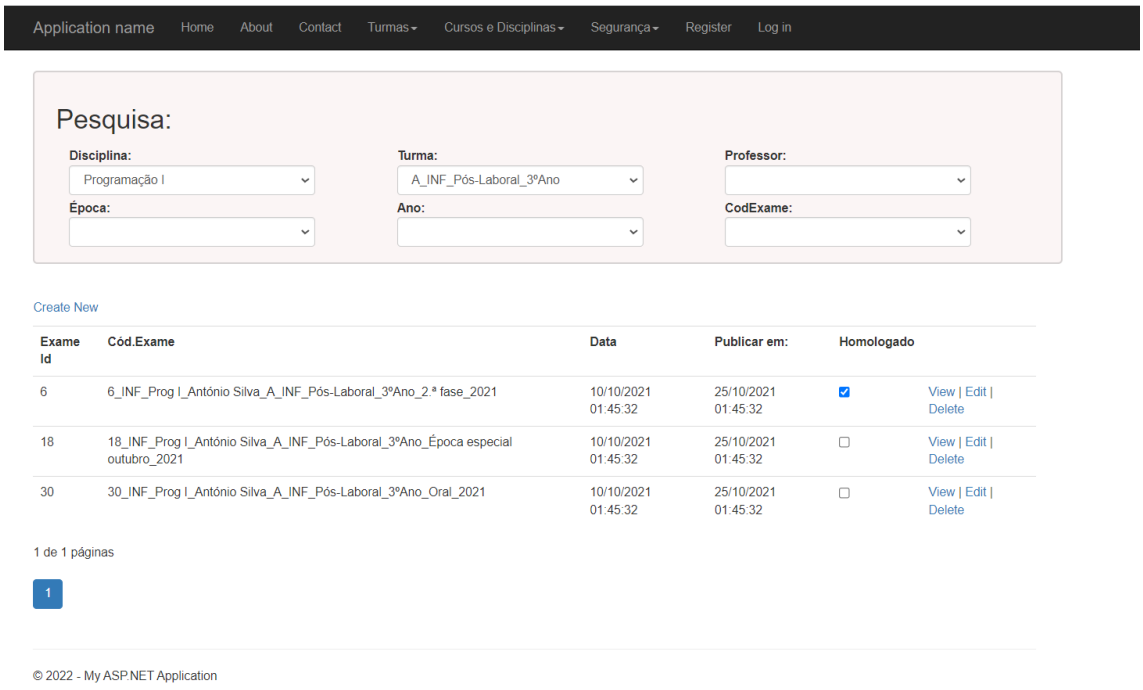


Figura 9-32 – View – Listar exames

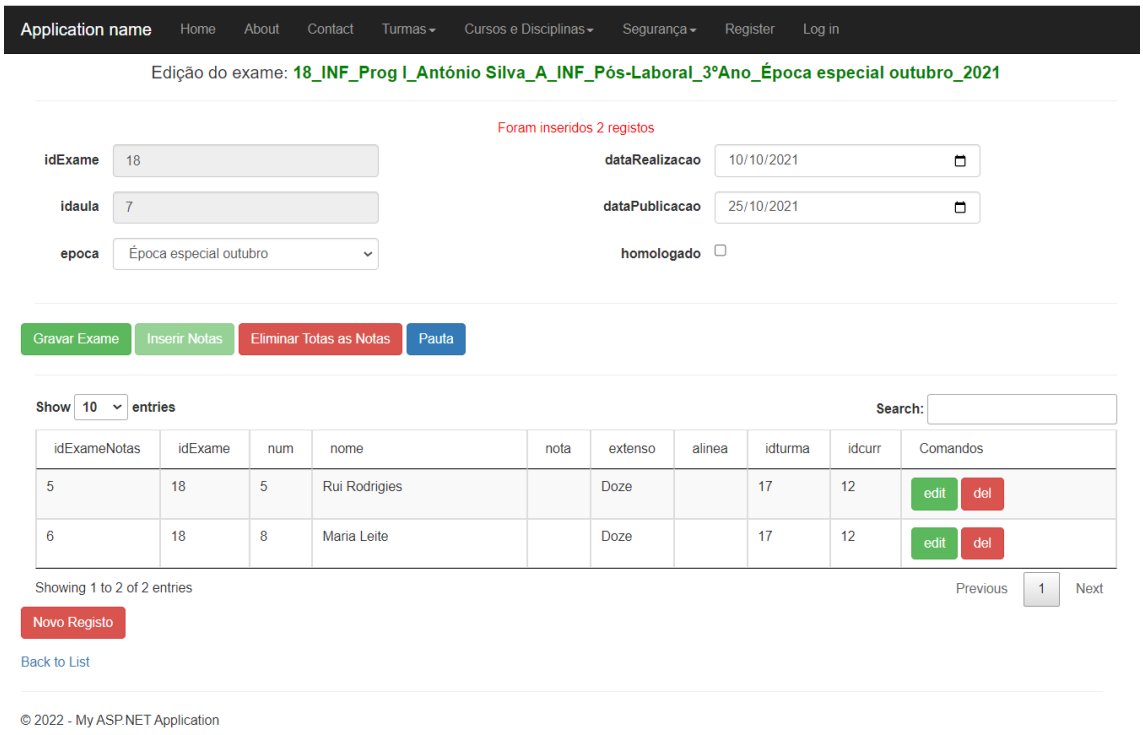
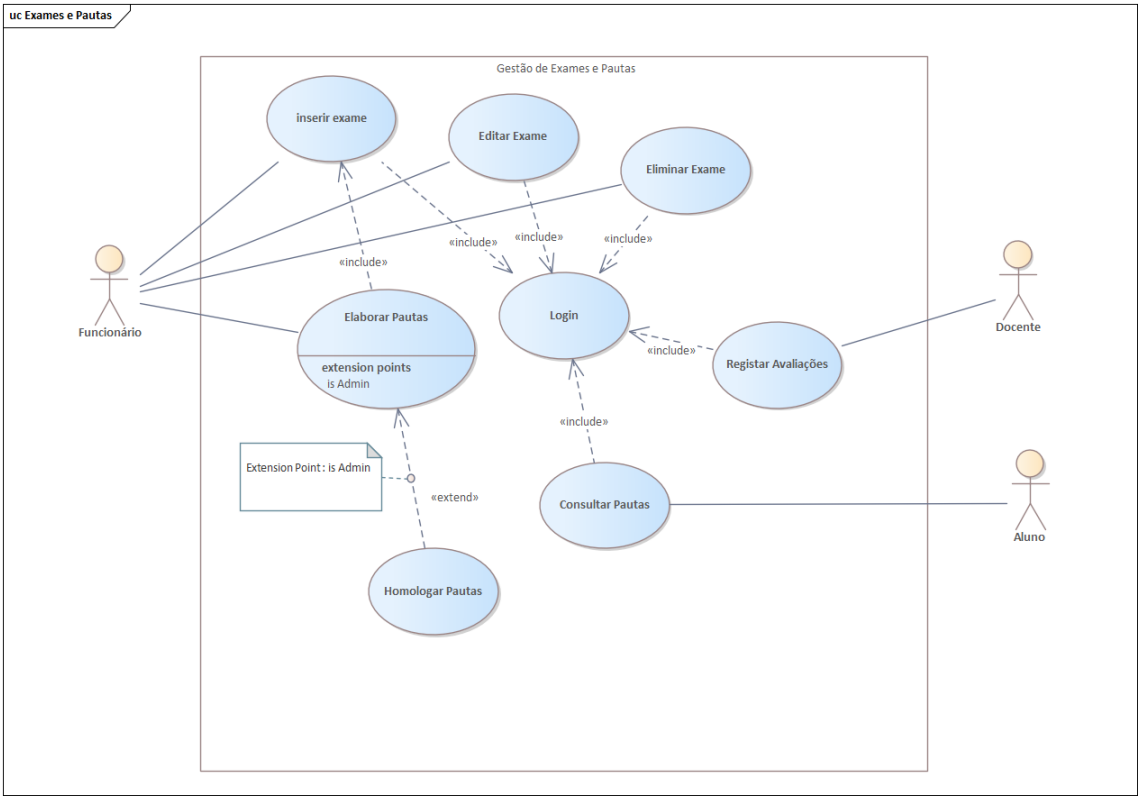


Figura 9-33 – View – Editar exame

Este "caso de uso" inclui as seguintes funcionalidades:

- ✓ Filtrar Exames;
- ✓ Inserir Exame;
- ✓ Editar exame;
  - Inserir, eliminar e editar as avaliações;
  - Editar pauta num ficheiro "excel";
- ✓ Eliminar exame;

9.2.1.2.4.10.1 Diagrama de caso de uso "Gerir Exame e Pautas"



9.2.1.2.4.10.2 <Descrição do caso do uso Gerir Exames e Pautas>

Identificação do Caso de Uso	7
Nome do caso de uso:	Gerir exames e pautas
Criado. Por:	José Neves
Data:	Novembro de 2021

Descrição:	A funcionalidades deste caso de uso compreendem: 1) inserir exame; 2) editar exame; 2a) inserir alunos na pauta; 2b) inserir notas e alíneas nos registos da pauta; 2c) Eliminar registos da pauta; 2b) emitir pauta num ficheiro excel; 3) Eliminar exame; 4) Homologar exame.

Caso de Uso Incluídos:	Login
Casos de uso extensão	Homologar pautas
Atores Primários:	Funcionário, docente, aluno
Atores Secundários:	Funcionário
Pré-Condições:	Todos os atores deverão estar autenticados. Os exames não poderão ser eliminados ou editados enquanto houver registos pendentes da entidade "ExameNotas"
Pós-Condições:	
Fluxo Principal	1) O funcionário insere novo exame; 2) O funcionário insere os registos da entidade "exameNotas" associados ao novo exame (implementa a pauta); 3) O funcionário elimina o exame; 4) O funcionário edita o exame; 5) O docente insere as avaliações dos exames que lhe dizem respeito; 6) O funcionário homologa as notas; 7) O aluno consulta os exames e respetivas avaliações.
Fluxo Alternativo	1B) Se funcionário não estiver autenticado é remetido para a página de Login e o fluxo principal passa para a faze 2; 1B.1) Se o utilizador não se autenticar termina o caso de uso; 3B) O funcionário não pode eliminar o exame se existirem registos de notas (exameNotas)pendentes; 4B) O funcionário não pode eliminar o exame se existirem registos de notas (entidade “exameNotas”) pendentes; 5B) O docente não pode registar avaliações nos exames que não lhe dizem respeito; 6B) O funcionário só pode homologar notas se for um administrador;

9.2.1.2.4.10.3 - Esquema relacional associado à entidade <Exame>

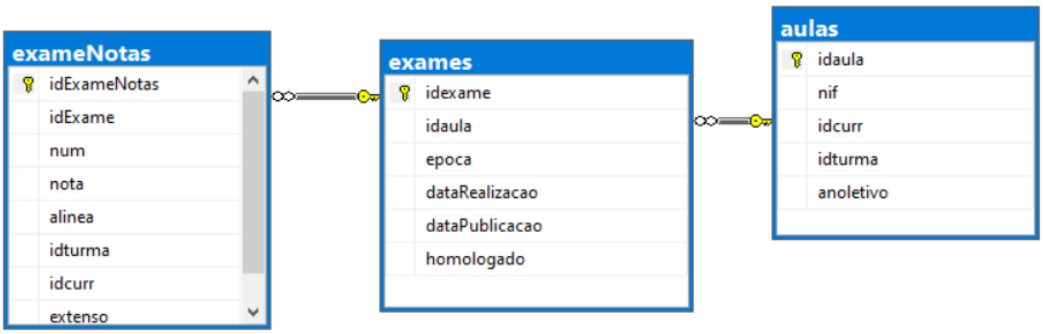


Figura 9-34 - Esquema relacional da entidade <exame>

De acordo com a figura uma aula está associada a vários exames e por sua vez cada exame está associado a vários registos da entidade <exameNotas>. Esta entidade reflete a avaliação de cada aluno no respetivo exame.

9.2.1.2.4.10.4 Diagrama de sequência do caso de uso Gerir Exame e Pautas

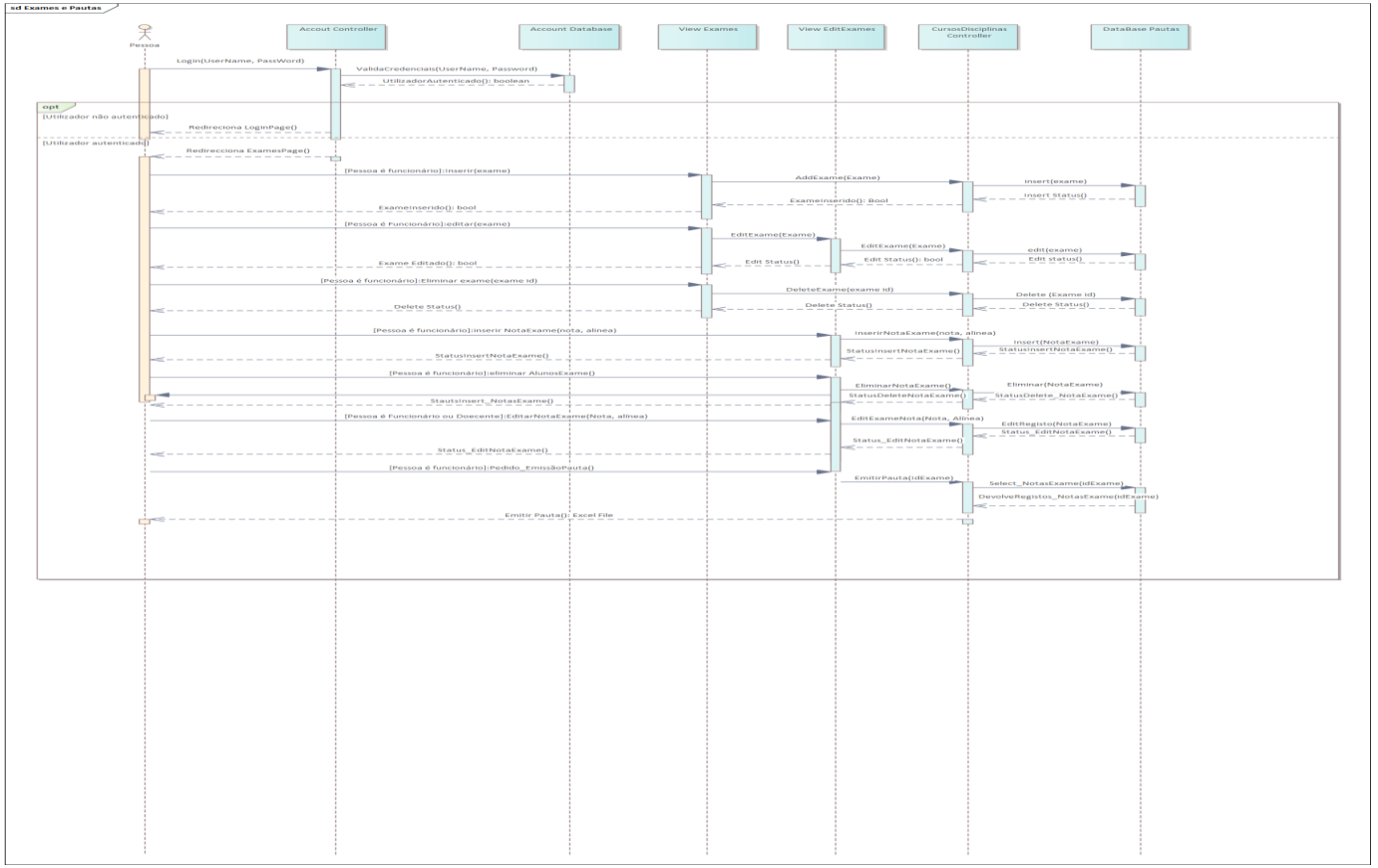


Figura 9-35 - Diagrama do caso de uso <Gerir exames e Pautas>

9.2.1.2.4.10.5 Implementação do caso de uso <Gerir Exames e Pautas – ANEXO_D>

9.2.1.2.4.10.6 Implementação na lógica do cliente do caso de uso <Anexo J & Anexo G >

9.2.1.2.4.11 Caso de uso <Gerir Acessos>

Este "caso de uso" compreende as seguintes funcionalidades:

- Registo de novas contas de utilizador;
- Controle de acessos e políticas de permissão;
- Edição de contas de utilizador com atribuição de "roles";

9.2.1.2.4.11.1 Diagrama Relacional associado a este caso de uso

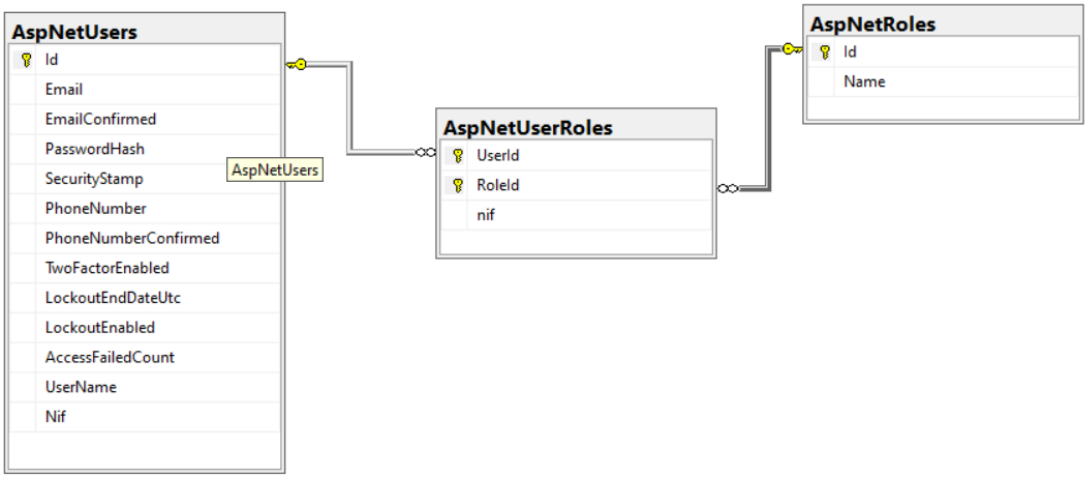


Figura 9-36 - Esquema relacional associado à entidade <AspNetUserRoles>

9.2.1.2.4.11.2 Diagrama de Classe do controller "Segurança"

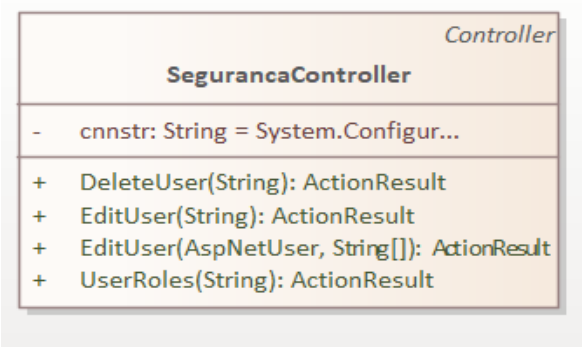


Figura 9-37 - Diagrama de classe o controller <Segurança>

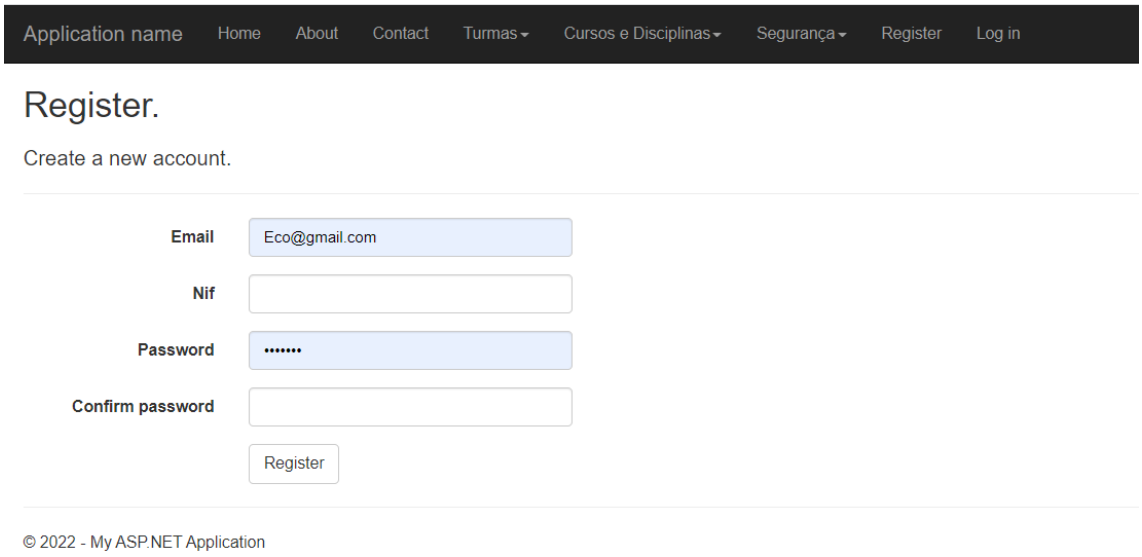


Figura 9-38 - View <Registo de novo utilizador.

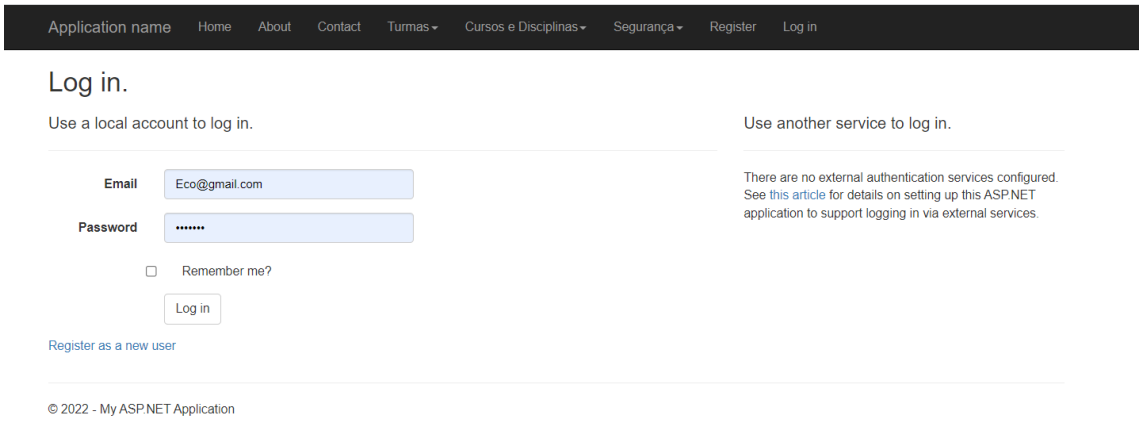


Figura 9-39 - View - Login de utilizadores

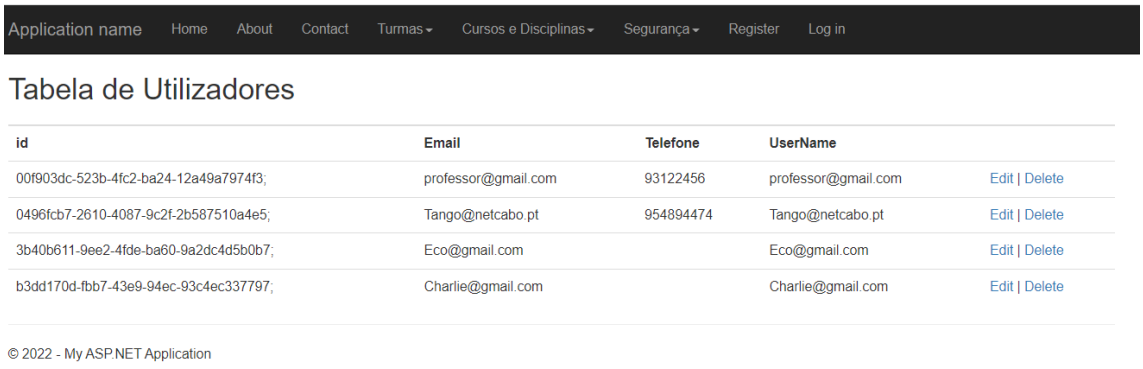


Figura 9-40 View <Listagem e Edição de Contas de Utilizador>

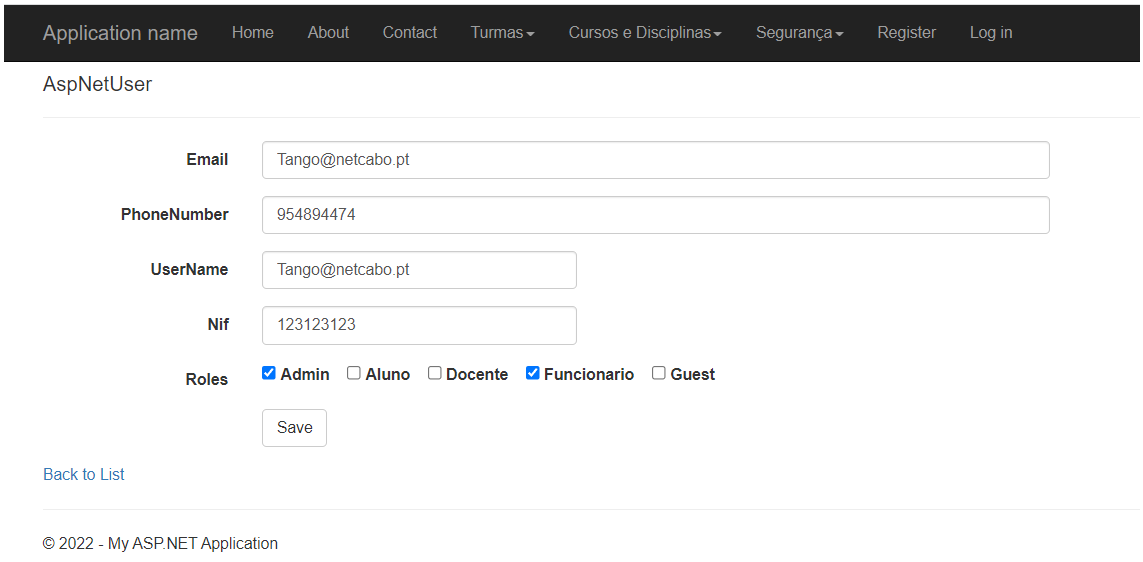
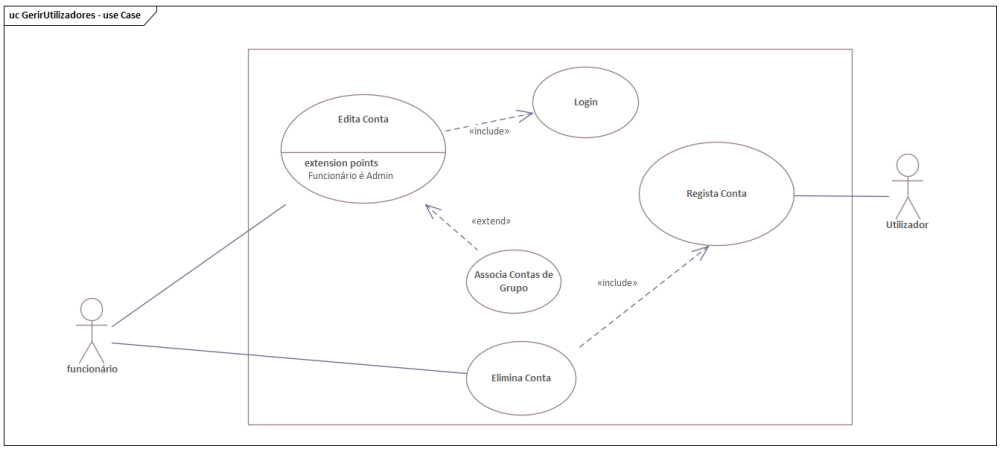


Figura 9-41 - View <Edição das Contas de Utilizador>

Esta view permite alterar os dados do utilizador e atribuir aos mesmos contas de Grupo.

9.2.1.2.4.11.3 <Diagrama de caso de uso – Gerir Utilizadores



9.2.1.2.4.11.4 <Descrição do caso de uso - Gerir Utilizadores>

Identificação do Caso de Uso	8
Use Case Name:	Gerir Utilizadores
Criado Por:	José Neves
Data:	Novembro de 2021

Descrição:	A funcionalidades deste caso de uso compreendem: 1) Os utilizadores Registam novas contas; 2) Os funcionários editam as Contas de Utilizador; 3) Os funcionários eliminam as contas de utilizador; 4) Os funcionários associam contas de grupo às contas de utilizador
Caso de Uso Incluídos:	Login
Casos de uso extensão	Associação de contas de grupo às contas individuais
Atores Primários:	Funcionário, utilizador geral
Atores Secundários:	Funcionário
Pré-Condições:	Os utilizadores terão que se registar
Pós-Condições:	

Fluxo Principal	1) O utilizador regista uma nova conta utilizador 2) O funcionário pode editar a conta atualizando os seguintes campos: - O funcionário edita NIF; - O funcionário edita o email; - O funcionário associa contas de grupo à conta de utilizador; - O funcionário edita o telefone; 3) O funcionário elimina contas de utilizador.
Fluxo Alternativo	2b) - Se o funcionário não pertencer à conta de Grupo "Admin" o Caso de Uso termina 3b) - Se o funcionário não pertencer à conta de Grupo "Admin" o Caso de Uso termina

9.2.1.2.4.11.5 Diagrama de sequência do caso de uso “Gerir Utilizadores”

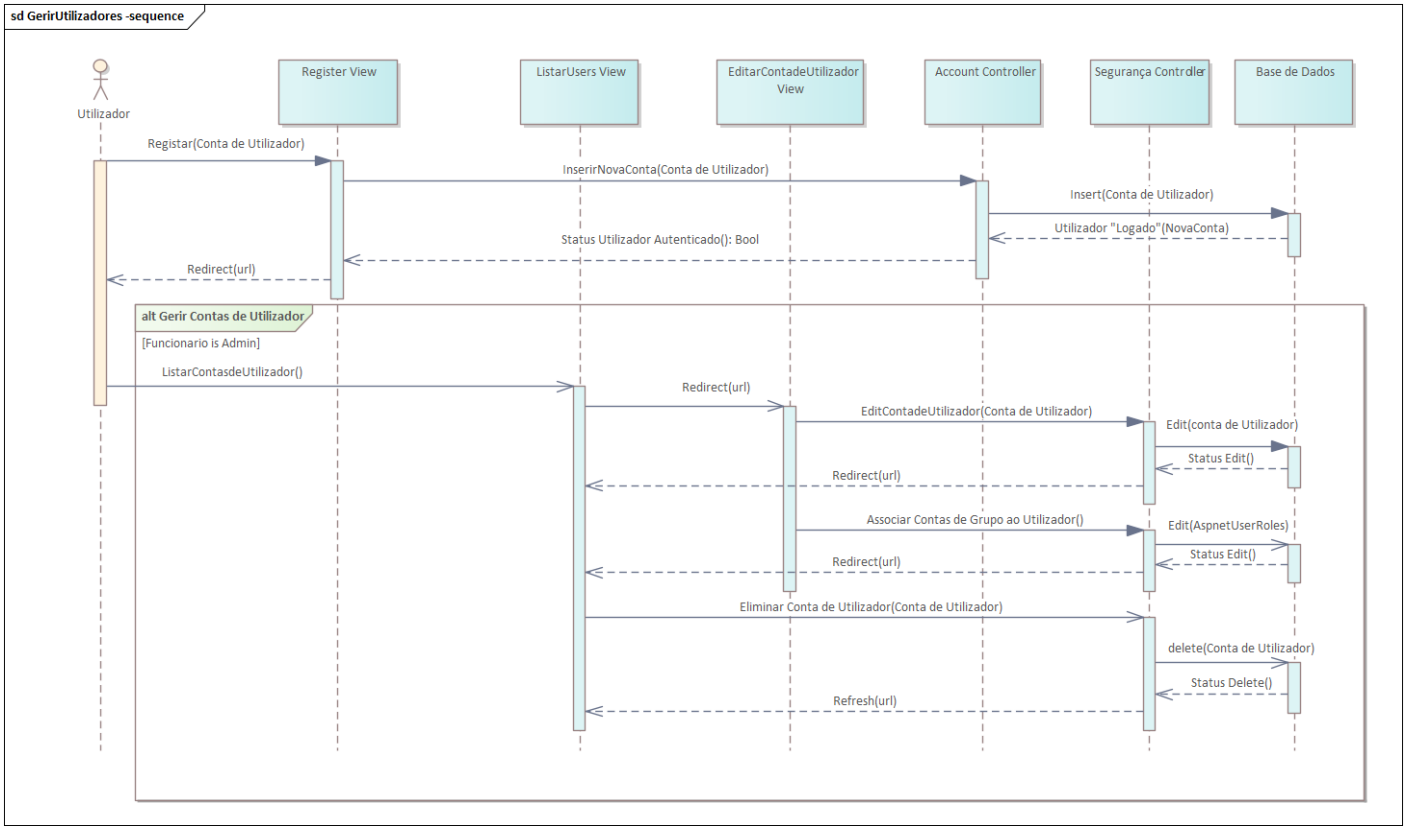


Figura 9-42 - Diagrama de sequência do caso de uso <Gerir Acessos>

9.2.1.2.4.11.6 Implementação do caso de uso <Gerir Acessos - Anexo E & Anexo L>

9.2.1.3 Modelo de Classes

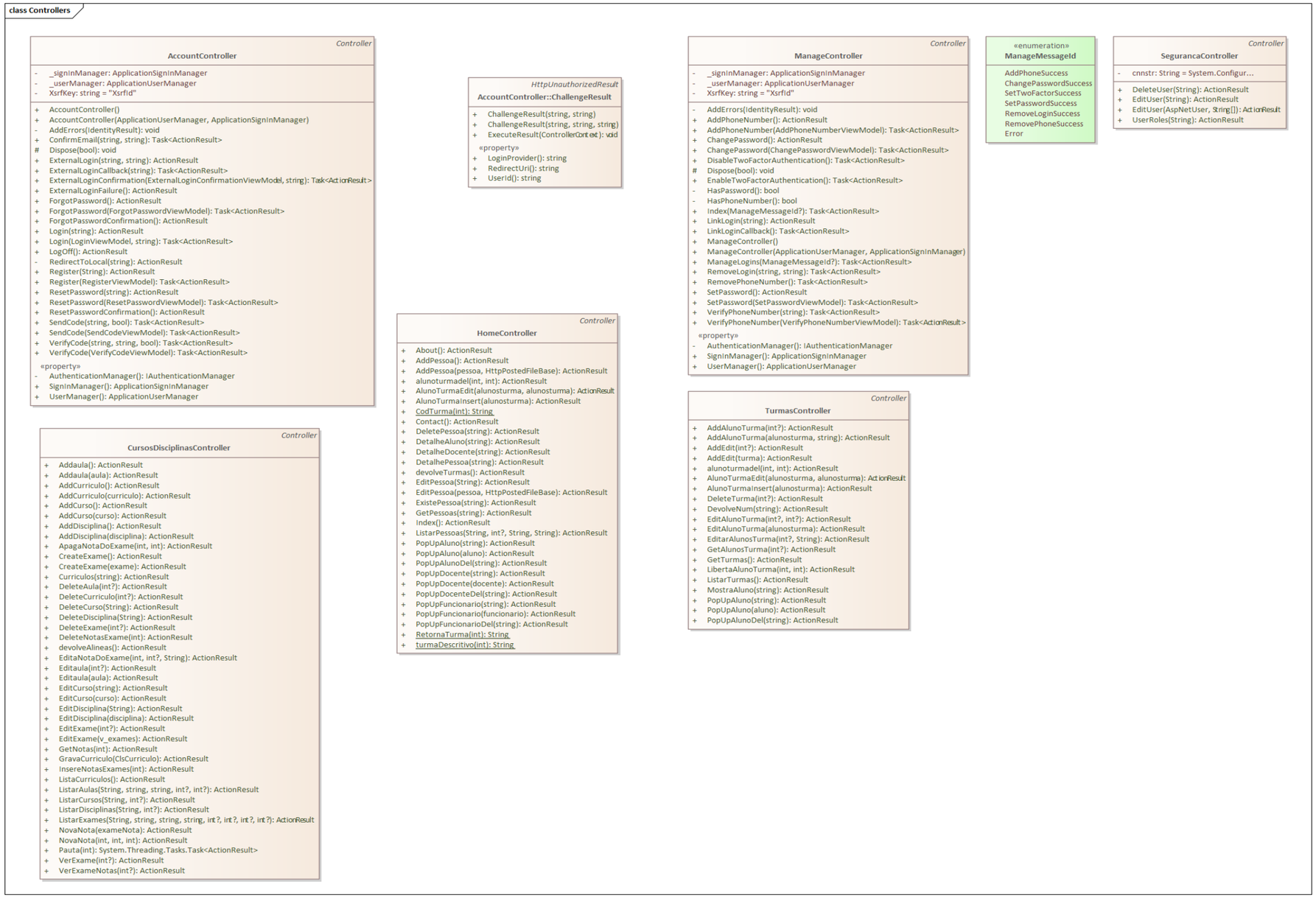


Figura 9-43 - Modelo de classes da aplicação

9.2.1.4 MODELO RELACIONAL

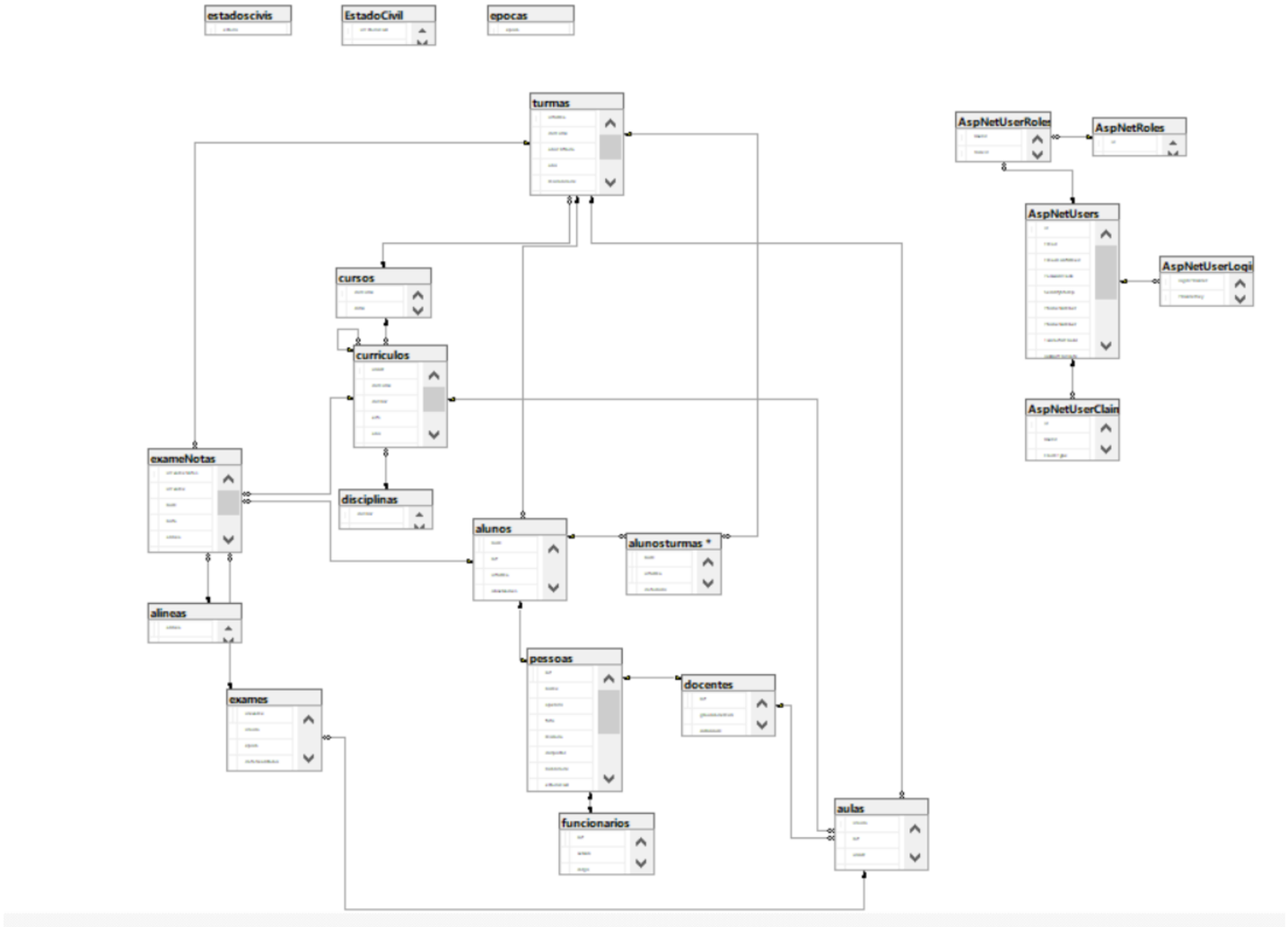


Figura 9-44 - Modelo Relacional da Base de Dados

9.2.2 Requisitos Não Funcionais

Relativamente aos requisitos não funcionais torna-se necessário que todos os intervenientes do sistema tenham alguma formação acerca das funcionalidades do sistema devendo ser fornecidos manuais que possibilitem a aprendizagem dum modo mais autónomo. É igualmente importante que todos os intervenientes tenham acesso à internet.

9.3 Principais Restrições de Projeto

Sendo um projeto piloto, mais um ensaio em contexto de uma prova de especialista, este projeto pretendeu essencialmente provar as capacidades do seu autor no que diz respeito a um perfil de especialista focado na área do desenvolvimento de aplicações e gestão de base de dados. Em situação normal o mesmo deveria ser desenvolvido por uma equipa multifacetada de programadores, gestores e analistas de sistemas o que colmataria a lacuna da escassez de tempo e da necessidade de aquisição de saberes e competências para a realização dum projeto desta envergadura.

Por outro lado, a falta de um teste decorrente da aplicação real a uma instituição de ensino e julgamento por parte de todos os intervenientes do sistema faz com que a sua eficiência e eficácia não tenham sido realmente postos à prova deixando antever que o mesmo deva carecer de outros desenvolvimentos e adaptações para cumprir os requisitos que nortearam a necessidade da sua implementação (Registo e consulta On-line das pautas de avaliação).

9.4 Requisitos Operacionais

Para a implementação deste projeto são necessários os seguintes requisitos:

Servidor SQL SERVER;

"Famework" do Aps.NET;

Alojamento para a aplicação em ASP.NET e base de Dados SQL SERVER;

Formação dos utilizadores sobre a operacionalização do sistema;

9.5 NORMAS

Este projeto deverá estar enquadrado com as seguintes normas:

Regulamento Geral da Proteção de dados;

Legislação para avaliação da aprendizagem no ensino superior;

Diretivas do regulamento interno da instituição

9.6 CONCLUSÕES

Face a um contexto hipotético de um estabelecimento de ensino superior onde o registo e a gestão das avaliações escolares dos alunos se processam de forma manual e pouco prática, procurou-se informatizar e agilizar estes processos.

Para o efeito foi encetada uma reflexão sobre o eventual funcionamento da instituição e deduzidas as seguintes necessidades:

- Os alunos terem permanente acesso "on-line" ao seu histórico de avaliações assim como o acesso atempado às pautas decorrentes dos exames;
- Permitir que os alunos solicitem on-line o seu certificado de habilitações e outros documentos julgados necessários tais como declarações de presença nos exames e etc.
- Facilitar aos docentes a capacidade de lançar as avaliações dos exames de que são responsáveis;
- Prestar apoio aos serviços de secretaria no planeamento, registo e organização dos exames;
- Agilizar os serviços da secretaria escolar referentes à emissão e à publicação das pautas decorrentes da realização dos exames;
- Agilizar, por parte da direção, a homologação e posterior publicação das pautas já preenchidas pelos docentes;
- Organizar e armazenar, de forma segura, o acervo dos registos de avaliações permitindo ainda, a partir deste suporte, gerir o histórico das avaliações dos alunos e a emissão dos documentos de certificação que dependam desses registos.

- Garantir que os processos inerentes ao suprimento destas necessidades sejam implementados de forma segura recorrendo-se a políticas de autorização e autenticação fiáveis;

Face a este diagnóstico o autor considerou implementar uma aplicação WEB que pudesse permitir a concretização dos objetivos preconizados. Entre as várias tecnologias disponíveis e que já fossem do seu domínio tais como: "PHP", "NODE JS" e "ASP.NET" o autor preferiu implementar o seu projeto em "ASP.NET" com recurso a base de dados "SQL SERVER". Esta escolha foi decidida em virtude desta tecnologia não só oferecer todos os recursos necessários como permitir fazê-lo de forma muito fiável, eficiente e segura. Complementa esta características com a enorme simplicidade e facilidade patenteadas na aquisição do domínio das suas ferramentas e ainda na facilidade de consecução de algumas soluções de que é exemplo paradigmático o sistema de segurança o qual requer um esforço mínimo para a sua implementação de modo eficiente e eficaz. A linguagem utilizada para desenvolver a aplicação foi o C#. Apesar de não ser uma "linguagem na moda" o autor considera-a uma das mais poderosas e sem dúvida alguma, aquela que melhor encaixa e plasma nos conceitos do paradigma de uma linguagem orientada a objetos.

Relativamente ao sistema de gestão de base de dados recorreu-se ao SQL SERVER uma vez que oferece todas as garantias de uma base de dados relacional sendo uma das ofertas mais implementadas no mercado profissional. Para além deste facto e sendo igualmente uma tecnologia Microsoft já vem naturalmente acoplada à "framework.NET".

Justificadas as escolhas para a realização do projeto importa salientar que foi implementada e conseguida a uma aplicação WEB onde foram satisfeitos a grande maioria dos objetivos inicialmente preconizados dos quais se destacam:

- Gestão dos atores do sistema e que compreendem: os docentes, alunos e funcionários;
- Gestão dos cursos ministrados na instituição;
- Gestão das disciplinas;
- Gestão dos currículos associados a cada curso;
- Gestão de exames;
- Elaboração e gestão das pautas de avaliação decorrentes desses exames;
- Lançamento de avaliações e alíneas nessas pautas;
- Homologação e publicação das pautas;
- Registo das pautas em ficheiros Excel;

Contudo, apesar dos resultados já obtidos julga-se necessária a implementação real da aplicação por forma a melhor aquilatar a eficácia e eficiência da mesma. Espera-se, ainda, que este processo possa propiciar a descoberta de novas necessidades e eventualmente a realização de novos desenvolvimentos que venham a ser vislumbrados pela lente da experiência em contexto real.

De momento, pode considerar-se, desde já a necessidade de acrescentar novos módulos que permitam o reaproveitamento do acervo de informação relativo às avaliações dos alunos. Neste sentido, poderia pensar-se na implementação das seguintes necessidades:

- Pedido e pagamento on-line do certificado de habilitações;
- Consulta do histórico de avaliações por parte de cada aluno garantindo que, de forma fácil e segura, o mesmo pudesse aceder em qualquer altura aos dados atualizados relativos ao seu percurso escolar;
- Um sistema de notificações qu

e permitisse de forma fácil e apelativa a troca de mensagens entre todos os intervenientes do sistema;

10 REFERÊNCIAS

Plínio Ventura. (2014, 28 de dezembro). *Caso de Uso – Include, Extend e Generalização* [Web log post]. Disponível a partir de https://www.ateomomento.com.br/caso-de-uso-include-extend-e-generalizacao/.

Fowler, M. (2007). *UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos*. 3ª.ed. Porto Alegre: Bookman.

Armstrong, D. (2005). *Pro ASP NET 2.0 Website Programming*. 1ªed. Apress

Sanderson, S. (2009). *Pro ASP.NET MVC framework*. Apress ; New York.

Anderson, R., Homer, A., Francis, B., Howard, R., Sussman, D., & Karli Watson. (2002). *Professional ASP. NET*. Wrox Press.

Macdonald, M. (2002). *ASP. NET : the complete reference*. Mcgraw-Hill/Osborne.

Bennett, S., Lunn, K., & Skelton, J. (2001). *Uml*. Mcgraw-Hill.

Perdita Stevens, & Pooley, R. J. (2006). *Using UML: software engineering with objects and components*. Addison-Wesley.

Conallen, J. (2003). *Building Web applications with UML*. Addison-Wesley.

Panttaja, J., Panttaja, M., & Prendergast, B. (1996). *The Microsoft SQL server survival guide*: [covers microsoft SQL server 6.5]. New York Wiley.

Bambara, J. J., & Allen, P. R. (2000). *SQL server developer's guide*. M & T Books.

Sharp, J. (2013). *Microsoft Visual C#2013 step by step*. O'reilly Media/Microsoft Press.

Michaels, F. (2009). Razor sharp. Kensington Pub.

Arlyn Hubbell. (2000). *Understanding Web development*. Prentice Hall.

11 ANEXO A

11.1 CONSULTA COM AS INSTRUÇÕES SQL PARA A IMPLEMENTAÇÃO DA BASE DE DADOS.

```
USE [master]
GO
/****** Object:  Database [pautas]    Script Date: 01/03/2022 18:34:40 ******/
CREATE DATABASE [pautas]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'pautas_mdf', FILENAME = N'D:\istec\especialista\pautas.mdf' , SIZE =
5120KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
( NAME = N'pautas_ldf', FILENAME = N'D:\istec\especialista\pautas.ldf' , SIZE =
5120KB , MAXSIZE = 2048GB , FILEGROWTH = 1024KB )
GO
ALTER DATABASE [pautas] SET COMPATIBILITY_LEVEL = 120
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [pautas].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [pautas] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [pautas] SET ANSI_NULLS OFF
GO
ALTER DATABASE [pautas] SET ANSI_PADDING OFF
GO
ALTER DATABASE [pautas] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [pautas] SET ARITHABORT OFF
GO
ALTER DATABASE [pautas] SET AUTO_CLOSE ON
GO
ALTER DATABASE [pautas] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [pautas] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [pautas] SET CURSOR_CLOSE_ON_COMMIT OFF
```

```
GO
ALTER DATABASE [pautas] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [pautas] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [pautas] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [pautas] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [pautas] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [pautas] SET  DISABLE_BROKER
GO
ALTER DATABASE [pautas] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [pautas] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [pautas] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [pautas] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [pautas] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [pautas] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [pautas] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [pautas] SET RECOVERY SIMPLE
GO
ALTER DATABASE [pautas] SET  MULTI_USER
GO
ALTER DATABASE [pautas] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [pautas] SET DB_CHAINING OFF
GO
ALTER DATABASE [pautas] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [pautas] SET TARGET_RECOVERY_TIME = 0 SECONDS
GO
ALTER DATABASE [pautas] SET DELAYED_DURABILITY = DISABLED
GO
USE [pautas]
GO
/****** Object:  UserDefinedFunction [dbo].[fn_apelido]    Script Date: 01/03/2022
18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
  CREATE function [dbo].[fn_apelido](@nome nvarchar(max)) returns nvarchar(max)
  as
  begin
```

```
   declare @apelido nvarchar(max) , @primeiro nvarchar(max), @ultimo nvarchar(max);
   declare @pos int =charindex(' ', @nome);
   if @pos >0
   Begin
      set @primeiro =LEFT(@nome,@pos-1);
         set @pos =CHARINDEX(' ' ,REVERSE(@nome))
         set @ultimo =RIGHT(@nome,@pos);
         set @apelido = @primeiro + @ultimo;
   End

   else set @apelido =@nome;
   return @apelido;

  end
GO
/****** Object:  UserDefinedFunction [dbo].[fn_ExisteDependencia]    Script Date:
01/03/2022 18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE function [dbo].[fn_ExisteDependencia](@nif as nvarchar(max))returns bit
AS
BEGIN
  declare @resp as bit;
  if exists(select nif from vwNifs where nif like @nif) set @resp= 1;
  else set @resp= 0;
  return @resp;
END
GO
/****** Object:  UserDefinedFunction [dbo].[fn_extenso]    Script Date: 01/03/2022
18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
 CREATE function [dbo].[fn_extenso](@nota int)returns nvarchar(max)
 AS
 BEGIN
  return
   case @nota
       when 0 then  'Zero'
          when 1 then  'Um'
          when 2 then  'Dois'
          when 3 then  'Três'
          when 4 then  'Quatro'
          when 5 then  'Cinco'
          when 6 then  'Seis'
          when 7 then  'Sete'
          when 8 then  'Oito'
          when 9 then  'Nove'
          when 10 then 'Dez'
          when 11 then 'Onze'
```

```
          when 12 then 'Doze'
          when 13 then 'Treze'
          when 14 then 'Catorze'
          when 15 then 'Quinze'
          when 16 then 'Dezasseis'
          when 17 then 'Dezassete'
          when 18 then 'Dezoito'
          when 19 then 'Dezanove'
          when 20 then 'Vinte'
          else ''
       end
 END
GO
/****** Object:  UserDefinedFunction [dbo].[fn_idcurr_exame]    Script Date:
01/03/2022 18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create function [dbo].[fn_idcurr_exame](@idexame int) returns int
AS
begin
       declare @idaula int;
       select @idaula = idaula from exames where idexame= @idexame;
       return (select idcurr from aulas where idaula=@idaula);
end
GO
/****** Object:  Table [dbo].[alineas]    Script Date: 01/03/2022 18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[alineas](
       [alinea] [char](6) NOT NULL,
       [descritivo] [nvarchar](256) NULL,
PRIMARY KEY CLUSTERED
(
       [alinea] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[alunos]    Script Date: 01/03/2022 18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[alunos](
       [num] [int] NOT NULL,
       [nif] [nvarchar](9) NOT NULL,
       [observacoes] [nvarchar](max) NULL,
       [dataEntrada] [date] NULL,
       [matriculaValidaAte] [date] NULL,
```

```
PRIMARY KEY CLUSTERED
(
	[num] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[alunosturmas]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[alunosturmas](
	[num] [int] NOT NULL,
	[idturma] [int] NOT NULL,
	[datainicio] [date] NOT NULL,
	[datafim] [date] NULL,
 CONSTRAINT [pk_alunos_turma] PRIMARY KEY CLUSTERED
(
	[num] ASC,
	[idturma] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[AspNetRoles]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetRoles](
	[Id] [nvarchar](128) NOT NULL,
	[Name] [nvarchar](256) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetRoles] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[AspNetUserClaims]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserClaims](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[UserId] [nvarchar](128) NOT NULL,
	[ClaimType] [nvarchar](max) NULL,
	[ClaimValue] [nvarchar](max) NULL,
```

```
 CONSTRAINT [PK_dbo.AspNetUserClaims] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[AspNetUserLogins]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserLogins](
	[LoginProvider] [nvarchar](128) NOT NULL,
	[ProviderKey] [nvarchar](128) NOT NULL,
	[UserId] [nvarchar](128) NOT NULL,
 CONSTRAINT [PK_dbo.AspNetUserLogins] PRIMARY KEY CLUSTERED
(
	[LoginProvider] ASC,
	[ProviderKey] ASC,
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[AspNetUserRoles]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUserRoles](
	[UserId] [nvarchar](128) NOT NULL,
	[RoleId] [nvarchar](128) NOT NULL,
	[nif] [nvarchar](9) NULL,
 CONSTRAINT [PK_dbo.AspNetUserRoles] PRIMARY KEY CLUSTERED
(
	[UserId] ASC,
	[RoleId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[AspNetUsers]    Script Date: 01/03/2022 18:34:40
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AspNetUsers](
	[Id] [nvarchar](128) NOT NULL,
	[Email] [nvarchar](256) NULL,
```

```
	[EmailConfirmed] [bit] NOT NULL,
	[PasswordHash] [nvarchar](max) NULL,
	[SecurityStamp] [nvarchar](max) NULL,
	[PhoneNumber] [nvarchar](max) NULL,
	[PhoneNumberConfirmed] [bit] NOT NULL,
	[TwoFactorEnabled] [bit] NOT NULL,
	[LockoutEndDateUtc] [datetime] NULL,
	[LockoutEnabled] [bit] NOT NULL,
	[AccessFailedCount] [int] NOT NULL,
	[UserName] [nvarchar](255) NULL,
	[Nif] [nvarchar](18) NULL,
 CONSTRAINT [PK_dbo.AspNetUsers] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[aulas]    Script Date: 01/03/2022 18:34:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[aulas](
	[idaula] [int] IDENTITY(1,1) NOT NULL,
	[nif] [nvarchar](9) NULL,
	[idcurr] [int] NULL,
	[idturma] [int] NULL,
	[anoletivo] [nvarchar](9) NULL,
PRIMARY KEY CLUSTERED
(
	[idaula] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[curriculos]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[curriculos](
	[idcurr] [int] IDENTITY(1,1) NOT NULL,
	[codCurso] [nvarchar](6) NULL,
	[codDisc] [nvarchar](12) NULL,
	[ects] [tinyint] NULL,
	[ano] [int] NULL,
	[semestre] [int] NULL,
	[codUnidade] [int] NULL,
	[precedencia] [int] NULL,
PRIMARY KEY CLUSTERED
(
```

```
	[idcurr] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[cursos]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[cursos](
	[codCurso] [nvarchar](6) NOT NULL,
	[curso] [nvarchar](120) NOT NULL,
	[nivel] [nvarchar](12) NULL,
PRIMARY KEY CLUSTERED
(
	[codCurso] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[disciplinas]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[disciplinas](
	[codDisc] [nvarchar](12) NOT NULL,
	[disciplina] [nvarchar](256) NULL,
PRIMARY KEY CLUSTERED
(
	[codDisc] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[docentes]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[docentes](
	[nif] [nvarchar](9) NOT NULL,
	[grauAcademico] [nvarchar](50) NULL,
	[curriculum] [nvarchar](max) NULL,
	[observacoes] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[nif] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

```
GO
/****** Object:  Table [dbo].[epocas]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[epocas](
	[epoca] [nvarchar](40) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[epoca] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[EstadoCivil]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[EstadoCivil](
	[idEstadoCivil] [nvarchar](20) NOT NULL,
	[estadoCivilTexto] [nvarchar](20) NULL,
PRIMARY KEY CLUSTERED
(
	[idEstadoCivil] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[estadoscivis]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[estadoscivis](
	[estado] [nvarchar](120) NOT NULL,
PRIMARY KEY CLUSTERED
(
	[estado] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[exameNotas]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[exameNotas](
```

```
	[idExameNotas] [int] IDENTITY(1,1) NOT NULL,
	[idExame] [int] NULL,
	[num] [int] NULL,
	[nota] [int] NULL,
	[alinea] [char](6) NULL,
	[idturma] [int] NULL,
	[idcurr] [int] NULL,
	[extenso]  AS ([dbo].[fn_extenso]([nota])),
PRIMARY KEY CLUSTERED
(
	[idExameNotas] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[exames]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[exames](
	[idexame] [int] IDENTITY(1,1) NOT NULL,
	[idaula] [int] NOT NULL,
	[epoca] [nvarchar](140) NOT NULL,
	[dataRealizacao] [datetime] NOT NULL,
	[dataPublicacao]  AS (dateadd(day,(15),[dataRealizacao])),
	[homologado] [bit] NULL,
PRIMARY KEY CLUSTERED
(
	[idexame] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[funcionarios]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[funcionarios](
	[nif] [nvarchar](9) NOT NULL,
	[servico] [nvarchar](120) NULL,
	[cargo] [nvarchar](120) NULL,
	[observacoes] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[nif] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[pessoas]    Script Date: 01/03/2022 18:34:41 ******/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[pessoas](
	[nif] [nvarchar](9) NOT NULL,
	[nome] [nvarchar](120) NOT NULL,
	[apelido]  AS ([dbo].[fn_apelido]([nome])),
	[foto] [nvarchar](256) NULL,
	[morada] [nvarchar](256) NULL,
	[codpostal] [nvarchar](8) NULL,
	[localidade] [nvarchar](256) NULL,
	[estadoCivil] [nvarchar](20) NULL,
	[sexo] [char](1) NULL,
	[estatuto] [nvarchar](20) NULL,
	[datanasc] [date] NULL,
	[idade]  AS (datediff(day,[datanasc],getdate())/(365.25)),
PRIMARY KEY CLUSTERED
(
	[nif] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[turmas]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[turmas](
	[idturma] [int] IDENTITY(1,1) NOT NULL,
	[codCurso] [nvarchar](6) NULL,
	[anoEntrada] [int] NULL,
	[ano]  AS (case when datepart(month,getdate())<(8) then
datepart(year,getdate())-[anoEntrada] else (datepart(year,getdate())-
[anoEntrada])+(1) end),
	[modalidade] [nvarchar](50) NULL,
	[UC] [bit] NOT NULL,
	[nome] [nvarchar](50) NULL,
	[codturma]  AS
(((((((([nome]+'_')+[codCurso])+'_')+[modalidade])+'_')+CONVERT([nvarchar](20),case
when datepart(month,getdate())<(8) then datepart(year,getdate())-[anoEntrada] else
(datepart(year,getdate())-[anoEntrada])+(1) end))+'ºAno'),
PRIMARY KEY CLUSTERED
(
	[idturma] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  View [dbo].[vw_aldocfunc]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
create view [dbo].[vw_aldocfunc]
AS
  select nif from docentes
  union
  select nif from alunos
  union
  select nif from funcionarios
GO
/****** Object:  View [dbo].[vw_alunosNaoAlocados]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE view  [dbo].[vw_alunosNaoAlocados]
as
select nif,nome from pessoas where (estatuto like 'Aluno' or estatuto is null or
estatuto='')
 and nif not in(select nif from vw_aldocfunc )
 union
 select p.nif,nome from pessoas p inner join alunos a on p.nif=a.nif and a.idturma
is null
GO
/****** Object:  View [dbo].[V_CodExames]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create View [dbo].[V_CodExames]
as
select e.idexame,
(cast(e.idexame as nvarchar(max)) + '_' +  t.codTurma + '_' + cur.codDisc +'_' +
p.apelido +'_' + e.epoca + '_' + CONVERT(VARCHAR(10), e.dataRealizacao, 111)  ) as
codExame
from exames e inner join aulas a on a.idaula=e.idaula
inner join curriculos cur on a.idcurr=cur.idcurr
inner join pessoas p on p.nif =a.nif
inner join turmas t on t.idturma=a.idturma
;
GO
/****** Object:  View [dbo].[v_exames]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE View [dbo].[v_exames]
as
SELECT e.* ,p.nif, p.apelido ,c.idcurr, c.codCurso, c.codDisc,t.idturma, t.codTurma
, (cast(e.idexame as nvarchar(6)) +'_' + c.codCurso +'_' + codDisc + '_' + p.apelido
```

```
+ '_' +t.codTurma+'_'+  ep.epoca +'_'+  + cast(year(e.dataRealizacao) as
nvarchar(4)))codExame
              ,iif(c.semestre=1,'1º Semestre','2º Semestre') semestre,
                      (
                case
                            when Month(dataRealizacao) between 1 and 8 then
cast(year(dataRealizacao)-1 as nvarchar(max)) + '/'  +cast(year(dataRealizacao) as
nvarchar(max))
                         else cast(year(dataRealizacao) as nvarchar(max)) + '/'  +
       cast(year(dataRealizacao)+1 as nvarchar(max))
                            end
                        )as anolectivo

                        from  exames e inner join aulas a on a.idaula=e.idaula
                    inner join pessoas p on p.nif=a.nif
                               inner join curriculos c on c.idcurr =a.idcurr
                               inner join turmas t on t.idturma=a.idturma
                               inner join epocas ep on ep.epoca=e.epoca

GO
/****** Object:  View [dbo].[V_exames_Ano]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create View [dbo].[V_exames_Ano]
AS
select distinct  year(dataRealizacao) ano from exames;
GO
/****** Object:  View [dbo].[v_notasExames]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE View [dbo].[v_notasExames]
as
select e.idExame, e.num, p.nome, iif(e.nota is null,'---',cast(e.nota as
nvarchar(max)))nota,iif(e.alinea is null,'',e.alinea)alinea, iif(e.nota is
null,'Faltou',e.extenso)extenso from exameNotas e inner join alunos a on e.num=a.num
inner join pessoas p on p.nif=a.nif
GO
/****** Object:  View [dbo].[v_professores]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create View [dbo].[v_professores]
as
select d.nif, p.nome from docentes d inner join pessoas p on d.nif=p.nif;
GO
```

```
/****** Object:  View [dbo].[v_seekaula]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE VIEW [dbo].[v_seekaula]
AS
select a.idaula, (cast(a.idaula as nvarchar(20)) +'_'+  c.codCurso +'_'+ c.codDisc +
'_' + t.codTurma + '_' + d.nome + '_' + anoletivo) as descritivo  from aulas a
     inner join pessoas d on a.nif=d.nif
     inner join curriculos c on c.idcurr=a.idcurr
        inner join turmas t on t.idturma=a.idturma;
GO
/****** Object:  View [dbo].[view_anoexames]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create View [dbo].[view_anoexames]
AS
select distinct  year(dataRealizacao) ano from exames;
GO
/****** Object:  View [dbo].[vw_alunos]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE view [dbo].[vw_alunos]
AS
select t.idturma, t.codTurma, a.num,p.nif,p.nome,p.foto, p.datanasc, floor(
DATEDIFF(day,datanasc,GETDATE())/365.25) as idade  from pessoas p inner join alunos
a on p.nif=a.nif
   inner join alunosturmas alt on a.num=alt.num
   inner join turmas t on t.idturma=alt.idturma where alt.datafim is null;
GO
/****** Object:  View [dbo].[vw_alunos_turma]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE view [dbo].[vw_alunos_turma]
as
select a.*, atr.idturma,atr.datainicio, atr.datafim from alunos a left join
alunosturmas atr on a.num=atr.num where atr.datafim is null
GO
/****** Object:  View [dbo].[Vw_alunoscurriculo]    Script Date: 01/03/2022 18:34:41
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE view [dbo].[Vw_alunoscurriculo] as
select atr.num, p.nome,cr.idcurr,t.idturma from cursos c inner join turmas t on
c.codCurso=t.codCurso
inner join curriculos  cr on c.codCurso=cr.codCurso
inner join alunosturmas atr on atr.idturma=t.idturma
inner join alunos a on a.num=atr.num
inner join pessoas p on p.nif=a.nif where atr.datafim is null ;
GO
/****** Object:  View [dbo].[vw_pessoas]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create view [dbo].[vw_pessoas]
AS  select nif,nome,foto,datanasc, idade, estatuto from pessoas


GO
/****** Object:  View [dbo].[vwNifs]    Script Date: 01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
Create View [dbo].[vwNifs]
as
 select nif from alunos
 union
 select nif from docentes
 union
 select nif from funcionarios;
GO
INSERT [dbo].[alineas] ([alinea], [descritivo]) VALUES (N'      ', N'')
INSERT [dbo].[alineas] ([alinea], [descritivo]) VALUES (N'a)    ', N'Oral')
INSERT [dbo].[alineas] ([alinea], [descritivo]) VALUES (N'b)    ', N'Sem  elementos
de Avaliação')
INSERT [dbo].[alineas] ([alinea], [descritivo]) VALUES (N'c)    ', N'Suspenso')
INSERT [dbo].[alunos] ([num], [nif], [observacoes], [dataEntrada],
[matriculaValidaAte]) VALUES (2, N'222333444', N'Aluno Novo', CAST(N'2021-09-01' AS
Date), CAST(N'2021-09-09' AS Date))
INSERT [dbo].[alunos] ([num], [nif], [observacoes], [dataEntrada],
[matriculaValidaAte]) VALUES (3, N'123456789', N'Inscrição da Maria Ontem',
CAST(N'2021-12-01' AS Date), CAST(N'2022-01-01' AS Date))
INSERT [dbo].[alunos] ([num], [nif], [observacoes], [dataEntrada],
[matriculaValidaAte]) VALUES (11, N'121121121', N'OK', CAST(N'2022-03-17' AS Date),
CAST(N'2023-06-01' AS Date))
INSERT [dbo].[alunos] ([num], [nif], [observacoes], [dataEntrada],
[matriculaValidaAte]) VALUES (22, N'333333333', N'Aluno com inscrição por pagar',
CAST(N'2022-01-01' AS Date), CAST(N'2023-09-09' AS Date))
INSERT [dbo].[alunos] ([num], [nif], [observacoes], [dataEntrada],
[matriculaValidaAte]) VALUES (77, N'545454545', NULL, CAST(N'2022-01-15' AS Date),
NULL)
INSERT [dbo].[alunosturmas] ([num], [idturma], [datainicio], [datafim]) VALUES (3,
1, CAST(N'2022-01-05' AS Date), CAST(N'2000-01-01' AS Date))
```

```
INSERT [dbo].[alunosturmas] ([num], [idturma], [datainicio], [datafim]) VALUES (3,
17, CAST(N'2022-01-05' AS Date), NULL)
INSERT [dbo].[alunosturmas] ([num], [idturma], [datainicio], [datafim]) VALUES (11,
1, CAST(N'0001-01-01' AS Date), NULL)
INSERT [dbo].[alunosturmas] ([num], [idturma], [datainicio], [datafim]) VALUES (22,
17, CAST(N'2022-01-09' AS Date), NULL)
INSERT [dbo].[alunosturmas] ([num], [idturma], [datainicio], [datafim]) VALUES (77,
17, CAST(N'2022-01-14' AS Date), CAST(N'2022-01-16' AS Date))
INSERT [dbo].[AspNetRoles] ([Id], [Name]) VALUES (N'admin', N'Admin')
INSERT [dbo].[AspNetRoles] ([Id], [Name]) VALUES (N'aluno', N'Aluno')
INSERT [dbo].[AspNetRoles] ([Id], [Name]) VALUES (N'docente', N'Docente')
INSERT [dbo].[AspNetRoles] ([Id], [Name]) VALUES (N'funcionario', N'Funcionario')
INSERT [dbo].[AspNetRoles] ([Id], [Name]) VALUES (N'guest', N'Guest')
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId], [nif]) VALUES (N'00f903dc-523b-
4fc2-ba24-12a49a7974f3', N'admin', NULL)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId], [nif]) VALUES (N'00f903dc-523b-
4fc2-ba24-12a49a7974f3', N'funcionario', NULL)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId], [nif]) VALUES (N'3b40b611-9ee2-
4fde-ba60-9a2dc4d5b0b7', N'admin', NULL)
INSERT [dbo].[AspNetUserRoles] ([UserId], [RoleId], [nif]) VALUES (N'3b40b611-9ee2-
4fde-ba60-9a2dc4d5b0b7', N'funcionario', NULL)
INSERT [dbo].[AspNetUsers] ([Id], [Email], [EmailConfirmed], [PasswordHash],
[SecurityStamp], [PhoneNumber], [PhoneNumberConfirmed], [TwoFactorEnabled],
[LockoutEndDateUtc], [LockoutEnabled], [AccessFailedCount], [UserName], [Nif])
VALUES (N'00f903dc-523b-4fc2-ba24-12a49a7974f3', N'professor@gmail.com', 0,
N'AN61VUupvBF5Dtp2tU8f4yVLsOQY7fSGbqBpk4tD5FuCGU3xgUpWUVhHHkoO/rP06Q==', N'c08dec98-
d57c-4636-92e4-a0c1cdc0f84c', N'93122456', 0, 0, NULL, 1, 0, N'professor@gmail.com',
N'987654321')
INSERT [dbo].[AspNetUsers] ([Id], [Email], [EmailConfirmed], [PasswordHash],
[SecurityStamp], [PhoneNumber], [PhoneNumberConfirmed], [TwoFactorEnabled],
[LockoutEndDateUtc], [LockoutEnabled], [AccessFailedCount], [UserName], [Nif])
VALUES (N'3b40b611-9ee2-4fde-ba60-9a2dc4d5b0b7', N'Eco@gmail.com', 0,
N'AMTb+eL5SIFaUzNcB2J7mK18c417g2gj5dOMmWvB9m0IQGFjr1hOL8bkiNr1ZiVZVg==', N'4a2485ba-
4595-401b-86df-a0a04e1091a4', NULL, 0, 0, NULL, 1, 0, N'Eco@gmail.com',
N'123123123')
INSERT [dbo].[AspNetUsers] ([Id], [Email], [EmailConfirmed], [PasswordHash],
[SecurityStamp], [PhoneNumber], [PhoneNumberConfirmed], [TwoFactorEnabled],
[LockoutEndDateUtc], [LockoutEnabled], [AccessFailedCount], [UserName], [Nif])
VALUES (N'b3dd170d-fbb7-43e9-94ec-93c4ec337797', N'Charlie@gmail.com', 0,
N'AJOvWnhkeI2gldxSUdf0OEMq0HRrECi9DYaljDMI2qINlBkHgE0aB3ELDa23qdxrtA==', N'26a83619-
239d-498a-9bdd-ade88f10e637', NULL, 0, 0, NULL, 1, 0, N'Charlie@gmail.com',
N'333444555')
SET IDENTITY_INSERT [dbo].[aulas] ON

INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (3,
N'188493514', 11, 1, N'2020/2021')
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (4,
N'188493514', 12, 1, N'2021/2022')
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (5,
N'188493514', 13, 1, N'2021/2022')
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (6,
N'188493514', 11, 17, N'2021/2022')
```

```
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (7,
N'188493514', 12, 17, N'2021/2022')
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (8,
N'188493514', 13, 17, N'2021/2022')
INSERT [dbo].[aulas] ([idaula], [nif], [idcurr], [idturma], [anoletivo]) VALUES (9,
N'188493514', 13, 1, N'2021/2022')
SET IDENTITY_INSERT [dbo].[aulas] OFF
SET IDENTITY_INSERT [dbo].[curriculos] ON

INSERT [dbo].[curriculos] ([idcurr], [codCurso], [codDisc], [ects], [ano],
[semestre], [codUnidade], [precedencia]) VALUES (11, N'INF', N'ICC', 1, 1, 1, 1111,
NULL)
INSERT [dbo].[curriculos] ([idcurr], [codCurso], [codDisc], [ects], [ano],
[semestre], [codUnidade], [precedencia]) VALUES (12, N'INF', N'Prog I', 2, 1, 1,
2222, NULL)
INSERT [dbo].[curriculos] ([idcurr], [codCurso], [codDisc], [ects], [ano],
[semestre], [codUnidade], [precedencia]) VALUES (13, N'INF', N'Tec.Int I', 3, 1, 1,
3333, 12)
SET IDENTITY_INSERT [dbo].[curriculos] OFF
INSERT [dbo].[cursos] ([codCurso], [curso], [nivel]) VALUES (N'INF', N'Informática',
N'Superior')
INSERT [dbo].[cursos] ([codCurso], [curso], [nivel]) VALUES (N'MULT', N'Multimédia',
N'Superior')
INSERT [dbo].[cursos] ([codCurso], [curso], [nivel]) VALUES (N'RSI', N'Redes e
Sistemas de Informação', N'Ctesp')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'AC', N'Arquitetura de
Computadores')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'ICC', N'Introdução à
Ciência dos Computadores')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'Mat I', N'Matemática
I')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'Prog I',
N'Programação I')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'RC I', N'Redes e
Comunicação I')
INSERT [dbo].[disciplinas] ([codDisc], [disciplina]) VALUES (N'Tec.Int I',
N'Tecnologias da Internet I')
INSERT [dbo].[docentes] ([nif], [grauAcademico], [curriculum], [observacoes]) VALUES
(N'188493514', N'Licenciatura', N'Licenciatura', NULL)
INSERT [dbo].[docentes] ([nif], [grauAcademico], [curriculum], [observacoes]) VALUES
(N'21212121', N'Licenciatura', N'Licenciatura', N'Excelente')
INSERT [dbo].[epocas] ([epoca]) VALUES (N'1.ª fase')
INSERT [dbo].[epocas] ([epoca]) VALUES (N'2.ª fase')
INSERT [dbo].[epocas] ([epoca]) VALUES (N'Época especial outubro')
INSERT [dbo].[epocas] ([epoca]) VALUES (N'Época especial setembro')
INSERT [dbo].[epocas] ([epoca]) VALUES (N'Oral')
INSERT [dbo].[EstadoCivil] ([idEstadoCivil], [estadoCivilTexto]) VALUES
(N'Casado(a)', N'Casado(a)')
INSERT [dbo].[EstadoCivil] ([idEstadoCivil], [estadoCivilTexto]) VALUES
(N'Divorciado(a)', N'Divorciado(a)')
INSERT [dbo].[EstadoCivil] ([idEstadoCivil], [estadoCivilTexto]) VALUES
(N'Solteiro(a)', N'Solteiro(a)')
```

```
INSERT [dbo].[EstadoCivil] ([idEstadoCivil], [estadoCivilTexto]) VALUES
(N'Viúvo(a)', N'Viúvo(a)')
INSERT [dbo].[estadoscivis] ([estado]) VALUES (N'casado(a)')
INSERT [dbo].[estadoscivis] ([estado]) VALUES (N'divorciado(a)')
INSERT [dbo].[estadoscivis] ([estado]) VALUES (N'solteiro(a)')
INSERT [dbo].[estadoscivis] ([estado]) VALUES (N'víuvo(a)')
SET IDENTITY_INSERT [dbo].[exameNotas] ON

INSERT [dbo].[exameNotas] ([idExameNotas], [idExame], [num], [nota], [alinea],
[idturma], [idcurr]) VALUES (118, 6, 3, NULL, NULL, 17, 12)
INSERT [dbo].[exameNotas] ([idExameNotas], [idExame], [num], [nota], [alinea],
[idturma], [idcurr]) VALUES (119, 6, 22, NULL, NULL, 17, 12)
INSERT [dbo].[exameNotas] ([idExameNotas], [idExame], [num], [nota], [alinea],
[idturma], [idcurr]) VALUES (120, 7, 3, NULL, NULL, 17, 13)
INSERT [dbo].[exameNotas] ([idExameNotas], [idExame], [num], [nota], [alinea],
[idturma], [idcurr]) VALUES (121, 7, 22, NULL, NULL, 17, 13)
SET IDENTITY_INSERT [dbo].[exameNotas] OFF
SET IDENTITY_INSERT [dbo].[exames] ON

INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (6, 7, N'2.ª fase', CAST(N'2021-10-10T01:45:32.057' AS DateTime), 1)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (7, 8, N'1.ª fase', CAST(N'2021-10-10T01:45:32.057' AS DateTime), 1)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (10, 5, N'2.ª fase', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (12, 7, N'2.ª fase', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (13, 8, N'2.ª fase', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (14, 3, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (15, 4, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (16, 5, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (17, 6, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (18, 7, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (19, 8, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (21, 4, N'Época especial setembro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (22, 5, N'Época especial setembro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
```

```
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (23, 6, N'Época especial setembro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (24, 7, N'Época especial setembro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (26, 3, N'Época especial outubro', CAST(N'2021-10-10T01:45:32.057' AS
DateTime), 1)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (27, 4, N'Oral', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (28, 5, N'Oral', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (29, 6, N'Oral', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (30, 7, N'Oral', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (31, 8, N'Oral', CAST(N'2021-10-10T01:45:32.057' AS DateTime), NULL)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (32, 3, N'1.ª fase', CAST(N'2022-02-06T00:00:00.000' AS DateTime), 0)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (33, 3, N'1.ª fase', CAST(N'2022-02-06T00:00:00.000' AS DateTime), 0)
INSERT [dbo].[exames] ([idexame], [idaula], [epoca], [dataRealizacao], [homologado])
VALUES (34, 3, N'2.ª fase', CAST(N'2022-02-06T00:00:00.000' AS DateTime), 0)
SET IDENTITY_INSERT [dbo].[exames] OFF
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'121121121', N'Benjamim',
N'/fotos/121121121.jfif', N'Rua do Alecrim', N'1200-345', N'Lourinhã',
N'Solteiro(a)', N'M', N'Aluno', CAST(N'1999-03-25' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'123456789', N'Maria da
Piedade', N'/fotos/123456789.jpg', N'Avenida Das Vitórias, 34 - 3ºdt.º', N'2000-
200', N'Braga', N'Solteiro(a)', N'M', N'Aluno', CAST(N'1996-12-11' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'188493514', N'António
Araújo Silva', N'/fotos/188493514.jpg', N'Praça Natália Correia nº 5 3ºDto', N'2720-
414', N'Damaia', N'Casado(a)', N'M', N'Docente', CAST(N'1963-05-10' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'21212121', N'José Manuel
Boturão das Neves', N'/fotos/21212121.jpg', N'Praça Natália Correia nº 5 3ºDto',
N'2720-414', N'Damaia', N'Solteiro(a)', N'M', N'Docente', CAST(N'1999-02-12' AS
Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'222333444', N'Rita Guerra',
N'/fotos/222333444.jpg', N'Rua da Rita', N'2300-400', N'Lousada', N'Solteiro(a)',
N'M', N'Aluno', CAST(N'1963-05-10' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'322322322', N'Cristina
Almeida', N'/fotos/322322322.jpg', N'Avenida Das Vitórias, 34 - 3ºdt.º', N'2000-
200', N'Braga', N'Solteiro(a)', N'M', N'Funcionário', CAST(N'1977-01-01' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'333333333', N'Rui
```

```
Rodrigies', N'/fotos/333333333.jpg', N'Rua do Catorze', N'1414-141', N'Xisto',
N'Divorciado(a)', N'M', N'Aluno', CAST(N'1991-03-06' AS Date))
INSERT [dbo].[pessoas] ([nif], [nome], [foto], [morada], [codpostal], [localidade],
[estadoCivil], [sexo], [estatuto], [datanasc]) VALUES (N'545454545', N'Maria Leite',
N'/fotos/545454545.jpg', N'Avenida Das Vitórias, 34 - 3ºdt.º', N'2000-200',
N'Braga', N'Solteiro(a)', N'M', N'Aluno', CAST(N'1977-01-01' AS Date))
SET IDENTITY_INSERT [dbo].[turmas] ON

INSERT [dbo].[turmas] ([idturma], [codCurso], [anoEntrada], [modalidade], [UC],
[nome]) VALUES (1, N'MULT', 2019, N'Laboral', 1, N'A')
INSERT [dbo].[turmas] ([idturma], [codCurso], [anoEntrada], [modalidade], [UC],
[nome]) VALUES (17, N'INF', 2019, N'Pós-Laboral', 1, N'A')
SET IDENTITY_INSERT [dbo].[turmas] OFF
SET ANSI_PADDING ON
GO
/****** Object:  Index [UQ__alunos__DF97D0F28077A3AC]    Script Date: 01/03/2022
18:34:41 ******/
ALTER TABLE [dbo].[alunos] ADD UNIQUE NONCLUSTERED
(
	[nif] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [RoleNameIndex]    Script Date: 01/03/2022 18:34:41 ******/
CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex] ON [dbo].[AspNetRoles]
(
	[Name] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [IX_UserId]    Script Date: 01/03/2022 18:34:41 ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserClaims]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [IX_UserId]    Script Date: 01/03/2022 18:34:41 ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserLogins]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
```

```
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [IX_RoleId]    Script Date: 01/03/2022 18:34:41 ******/
CREATE NONCLUSTERED INDEX [IX_RoleId] ON [dbo].[AspNetUserRoles]
(
	[RoleId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [IX_UserId]    Script Date: 01/03/2022 18:34:41 ******/
CREATE NONCLUSTERED INDEX [IX_UserId] ON [dbo].[AspNetUserRoles]
(
	[UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [UQ__cursos__B9A3FF626157B56E]    Script Date: 01/03/2022
18:34:41 ******/
ALTER TABLE [dbo].[cursos] ADD UNIQUE NONCLUSTERED
(
	[curso] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [UQ__discipli__E3ED214B5EA57384]    Script Date: 01/03/2022
18:34:41 ******/
ALTER TABLE [dbo].[disciplinas] ADD UNIQUE NONCLUSTERED
(
	[disciplina] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [UQ__estadosc__2294F42E78ABE1F5]    Script Date: 01/03/2022
18:34:41 ******/
ALTER TABLE [dbo].[EstadoCivil] ADD UNIQUE NONCLUSTERED
(
	[estadoCivilTexto] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
```

```
GO
/****** Object:  Index [uc_examenotas]    Script Date: 01/03/2022 18:34:41 ******/
ALTER TABLE [dbo].[exameNotas] ADD  CONSTRAINT [uc_examenotas] UNIQUE NONCLUSTERED
(
	[idExame] ASC,
	[num] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
/****** Object:  Index [UnicoNumExame]    Script Date: 01/03/2022 18:34:41 ******/
ALTER TABLE [dbo].[exameNotas] ADD  CONSTRAINT [UnicoNumExame] UNIQUE NONCLUSTERED
(
	[num] ASC,
	[idExame] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
GO
ALTER TABLE [dbo].[turmas] ADD  DEFAULT (datepart(year,getdate())) FOR [anoEntrada]
GO
ALTER TABLE [dbo].[turmas] ADD  DEFAULT ((0)) FOR [UC]
GO
ALTER TABLE [dbo].[alunos]  WITH CHECK ADD  CONSTRAINT [fkalunopessoa] FOREIGN
KEY([nif])
REFERENCES [dbo].[pessoas] ([nif])
ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[alunos] CHECK CONSTRAINT [fkalunopessoa]
GO
ALTER TABLE [dbo].[alunosturmas]  WITH CHECK ADD  CONSTRAINT
[fk_alunos_alunosTurmas] FOREIGN KEY([num])
REFERENCES [dbo].[alunos] ([num])
GO
ALTER TABLE [dbo].[alunosturmas] CHECK CONSTRAINT [fk_alunos_alunosTurmas]
GO
ALTER TABLE [dbo].[alunosturmas]  WITH CHECK ADD  CONSTRAINT
[fk_turmas_alunosTurmas] FOREIGN KEY([idturma])
REFERENCES [dbo].[turmas] ([idturma])
GO
ALTER TABLE [dbo].[alunosturmas] CHECK CONSTRAINT [fk_turmas_alunosTurmas]
GO
ALTER TABLE [dbo].[AspNetUserClaims]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserClaims_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserClaims] CHECK CONSTRAINT
[FK_dbo.AspNetUserClaims_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[AspNetUserLogins]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserLogins_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
```

```
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserLogins] CHECK CONSTRAINT
[FK_dbo.AspNetUserLogins_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[AspNetUserRoles]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetRoles_RoleId] FOREIGN KEY([RoleId])
REFERENCES [dbo].[AspNetRoles] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetRoles_RoleId]
GO
ALTER TABLE [dbo].[AspNetUserRoles]  WITH CHECK ADD  CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetUsers_UserId] FOREIGN KEY([UserId])
REFERENCES [dbo].[AspNetUsers] ([Id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT
[FK_dbo.AspNetUserRoles_dbo.AspNetUsers_UserId]
GO
ALTER TABLE [dbo].[aulas]  WITH CHECK ADD FOREIGN KEY([idcurr])
REFERENCES [dbo].[curriculos] ([idcurr])
GO
ALTER TABLE [dbo].[aulas]  WITH CHECK ADD FOREIGN KEY([idturma])
REFERENCES [dbo].[turmas] ([idturma])
GO
ALTER TABLE [dbo].[aulas]  WITH CHECK ADD FOREIGN KEY([nif])
REFERENCES [dbo].[docentes] ([nif])
GO
ALTER TABLE [dbo].[curriculos]  WITH CHECK ADD FOREIGN KEY([codCurso])
REFERENCES [dbo].[cursos] ([codCurso])
GO
ALTER TABLE [dbo].[curriculos]  WITH CHECK ADD FOREIGN KEY([codDisc])
REFERENCES [dbo].[disciplinas] ([codDisc])
GO
ALTER TABLE [dbo].[curriculos]  WITH CHECK ADD  CONSTRAINT [fkprecedencia] FOREIGN
KEY([precedencia])
REFERENCES [dbo].[curriculos] ([idcurr])
GO
ALTER TABLE [dbo].[curriculos] CHECK CONSTRAINT [fkprecedencia]
GO
ALTER TABLE [dbo].[docentes]  WITH CHECK ADD  CONSTRAINT [fkdocentepessoa] FOREIGN
KEY([nif])
REFERENCES [dbo].[pessoas] ([nif])
ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[docentes] CHECK CONSTRAINT [fkdocentepessoa]
GO
ALTER TABLE [dbo].[exameNotas]  WITH CHECK ADD FOREIGN KEY([alinea])
REFERENCES [dbo].[alineas] ([alinea])
GO
ALTER TABLE [dbo].[exameNotas]  WITH CHECK ADD FOREIGN KEY([idcurr])
```

```
REFERENCES [dbo].[curriculos] ([idcurr])
GO
ALTER TABLE [dbo].[exameNotas]  WITH CHECK ADD FOREIGN KEY([idExame])
REFERENCES [dbo].[exames] ([idexame])
GO
ALTER TABLE [dbo].[exameNotas]  WITH CHECK ADD  CONSTRAINT
[fk_alunosturmas_examenotas] FOREIGN KEY([num], [idturma])
REFERENCES [dbo].[alunosturmas] ([num], [idturma])
GO
ALTER TABLE [dbo].[exameNotas] CHECK CONSTRAINT [fk_alunosturmas_examenotas]
GO
ALTER TABLE [dbo].[exames]  WITH CHECK ADD FOREIGN KEY([idaula])
REFERENCES [dbo].[aulas] ([idaula])
GO
ALTER TABLE [dbo].[funcionarios]  WITH CHECK ADD  CONSTRAINT [fkfuncionariopessoa]
FOREIGN KEY([nif])
REFERENCES [dbo].[pessoas] ([nif])
ON UPDATE CASCADE
GO
ALTER TABLE [dbo].[funcionarios] CHECK CONSTRAINT [fkfuncionariopessoa]
GO
ALTER TABLE [dbo].[turmas]  WITH CHECK ADD FOREIGN KEY([codCurso])
REFERENCES [dbo].[cursos] ([codCurso])
GO
ALTER TABLE [dbo].[curriculos]  WITH CHECK ADD CHECK  (([semestre]>=(1) AND
[semestre]<=(2)))
GO
ALTER TABLE [dbo].[curriculos]  WITH CHECK ADD CHECK  (([ano]>=(1) AND [ano]<=(3)))
GO
ALTER TABLE [dbo].[cursos]  WITH CHECK ADD CHECK  (([nivel]='Superior' OR
[nivel]='CTesp'))
GO
ALTER TABLE [dbo].[exameNotas]  WITH CHECK ADD CHECK  (([nota]>=(0) AND
[nota]<=(20)))
GO
ALTER TABLE [dbo].[turmas]  WITH CHECK ADD CHECK  (([anoEntrada]>=(1970) AND
[anoEntrada]<=(2050)))
GO
/****** Object:  StoredProcedure [dbo].[apagaEstatuto]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
 create proc [dbo].[apagaEstatuto](@nif nvarchar(max))
 AS
 BEGIN
    delete from alunos where nif like @nif;
       delete from docentes where nif like @nif;
    delete from funcionarios where nif like @nif;

 END
GO
```

```
/****** Object:  StoredProcedure [dbo].[sp_aluno]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE proc [dbo].[sp_aluno](
 @num int
) AS
BEGIN
 select * from alunos where num=@num;
END
GO
/****** Object:  StoredProcedure [dbo].[sp_alunos_livres]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE proc [dbo].[sp_alunos_livres]
as
begin
       select nif ,nome from pessoas where nif in(
       select nif from pessoas where estatuto Like 'Aluno' and nif not in(select nif
from alunos)
       union all
       select  nif from alunos where num not in (select  num from alunosturmas where
datafim is null));
end
GO
/****** Object:  StoredProcedure [dbo].[sp_del_notasExame]    Script Date:
01/03/2022 18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[sp_del_notasExame](@idexame int,@total int out)
AS
begin

  select  @total=count(*)from exameNotas where idExame=@idexame;
  delete from exameNotas where idExame=@idexame;
end
GO
/****** Object:  StoredProcedure [dbo].[sp_getPessoa]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
create proc [dbo].[sp_getPessoa](@nif nvarchar(max) )
AS
BEGIN
```

```
  select  * from pessoas where nif like  @nif;
END
GO
/****** Object:  StoredProcedure [dbo].[sp_notasExame]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[sp_notasExame](
  @idexame int,
  @total int out
)
AS
begin
  declare @idturma int, @idaula int, @idcurr int;
  select @idaula=idaula from exames where idexame=@idexame;
  select @idturma=idturma, @idcurr=idcurr from aulas where idaula=@idaula;
  insert into exameNotas(idExame,num,idturma,idcurr)select
@idexame,num,@idturma,@idcurr from alunosturmas where idturma=@idturma and datafim
is null;
  select @total=count(*) from  exameNotas where idExame=@idexame;
  select @total;
end
GO
/****** Object:  StoredProcedure [dbo].[sp_seek_exames]    Script Date: 01/03/2022
18:34:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE procedure [dbo].[sp_seek_exames](
              @nif as nvarchar(20) =NULL,
              @idturma as int =NULL,
              @codDisc nvarchar(50)=NULL ,
              @codCurso nvarchar(50)=NULL ,
              @epoca nvarchar(250)=NULL,
              @ano int =NULL,
              @idexame int =NULL
 )
AS
Begin
       (((((
       select  * from v_exames where idaula in(select idaula from aulas where nif=
@nif or  @nif is null )
       intersect
       select  * from v_exames where idaula in(select idaula from aulas where
idturma=@idturma OR  @idturma is null)
       )
       intersect
       select  * from v_exames where idaula in (select idaula from aulas where
idcurr in(select idcurr from curriculos where codDisc = @codDisc  or @codDisc is
NULL )  )
```

```
	)
	intersect
	select  * from v_exames where idaula in (select idaula from aulas where
idcurr in(select idcurr from curriculos where codCurso like @codCurso or  @codCurso
is NULL  )  )
	)
	intersect
	select  * from v_exames where epoca like @epoca or @epoca is NULL
	)
	intersect
	select  * from v_exames where year(dataRealizacao) =@ano or @ano is NULL
	)
	intersect
	select  * from v_exames where idexame = @idexame or @idexame is null
END;
GO
USE [master]
GO
ALTER DATABASE [pautas] SET  READ_WRITE
GO
```

12 ANEXO B <HOME CONTROLLER>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using pautas.Models;
using Newtonsoft.Json;

namespace pautas.Controllers
{
   [Authorize(Roles = "Funcionario,Docente")]
   public class TurmasController : Controller
   {
      public ActionResult AlunoTurmaInsert(alunosturma alunoturma) {
         try
         {
            using (pautasEntities db = new pautasEntities())
            {
               alunosturma nova = new alunosturma() { num = alunoturma.num, idturma =
                     alunoturma.idturma, datainicio = alunoturma.datainicio, datafim =
                                                                  alunoturma.datafim };
               db.alunosturmas.Add(nova);
               db.SaveChanges();
               var turmasaluno = (from at in db.alunosturmas.Where(x => x.num ==
                                                                  alunoturma.num)
                        join t in db.turmas on at.idturma equals t.idturma
                        select new { num = at.num, idturma = at.idturma, t.codturma,
                        datainicio = at.datainicio, datafim = at.datafim }).ToList();
               return Json(new { count = turmasaluno.Count(), turmasAluno = turmasaluno, msg
                        = "ok" }, JsonRequestBehavior.AllowGet);
```

```
            }

        }
        catch (Exception erro)
        {
            using (pautasEntities db = new pautasEntities())
            {
                var turmasaluno = (from at in db.alunosturmas.Where(x => x.num ==
                                                                    alunoturma.num)
                                   join t in db.turmas on at.idturma equals t.idturma
                                   select new { num = at.num, idturma = at.idturma, t.codturma, datainicio
                                                = at.datainicio, datafim = at.datafim }).ToList();
                                   return Json(new { count = -1, turmasAluno = turmasaluno, msg =
                                    erro.Message }, JsonRequestBehavior.AllowGet);

            }
        }
    }
    public ActionResult AlunoTurmaEdit(alunosturma newAluno, alunosturma oldAluno)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                alunosturma velho = new alunosturma() { num = oldAluno.num, idturma =
oldAluno.idturma, datainicio = oldAluno.datainicio, datafim = oldAluno.datafim };
                alunosturma nova = new alunosturma() { num = newAluno.num, idturma =
newAluno.idturma, datainicio = newAluno.datainicio, datafim = newAluno.datafim };
                alunosturma este = db.alunosturmas.Where(x => x.num == velho.num &&
x.idturma == velho.idturma && x.datainicio == velho.datainicio).FirstOrDefault();
                if (este != null)
                {
                    db.alunosturmas.Remove(este);
                    db.SaveChanges();
                }
                db.alunosturmas.Add(nova);
                db.SaveChanges();
```

```
                var turmasaluno = (from at in db.alunosturmas.Where(x => x.num == nova.num)
                                   join t in db.turmas on at.idturma equals t.idturma
                                   select new { num = at.num, idturma = at.idturma, t.codturma,
datainicio = at.datainicio, datafim = at.datafim }).ToList();
                return Json(new { count = turmasaluno.Count(), turmasAluno = turmasaluno, msg
= "Registo Editado com sucesso" }, JsonRequestBehavior.AllowGet);   }

        catch (Exception erro)
            {
              using (pautasEntities db = new pautasEntities())
              {
                var turmasaluno = (from at in db.alunosturmas.Where(x => x.num ==
newAluno.num)
                                   join t in db.turmas on at.idturma equals t.idturma
                                   select new { num = at.num, idturma = at.idturma, t.codturma,
datainicio = at.datainicio, datafim = at.datafim }).ToList();

                return Json(new { count = 0, turmasAluno = turmasaluno, msg = erro.Message },
JsonRequestBehavior.AllowGet);
              }
            }
        }
        public ActionResult alunoturmadel(int idturma, int num)
        {

          try
          {
            using (pautasEntities db = new pautasEntities())
            {
              alunosturma item = db.alunosturmas.Where(x => x.idturma == idturma && x.num
== num).FirstOrDefault();
              if (item != null)
              {
                db.alunosturmas.Remove(item);
                db.SaveChanges();
                return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);
              }
```

```
            }
        }
        catch (Exception e)     {
            return Json(new { msg = e.Message }, JsonRequestBehavior.AllowGet);
        }
        return Json(new { msg = "erro" }, JsonRequestBehavior.AllowGet);
    }

    public ActionResult PopUpAlunoDel(string nif)   {
        using (pautasEntities db = new pautasEntities())
        {
            try
            {
                aluno a = db.alunos.Where(x => x.nif == nif).SingleOrDefault();
                if (a != null)
                {
                    db.alunos.Remove(a);
                    pessoa p = db.pessoas.Find(nif);
                    if (p != null) p.estatuto = "";
                    db.SaveChanges();
                    return Json(new { msg = "Aluno Eliminado com sucesso" },
JsonRequestBehavior.AllowGet);
                }
                else return Json(new { msg = "Aluno não existe" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception erro)
            {
                return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
            }


        }

    }
    public ActionResult PopUpAluno(string nif)
```

```
{
   using (pautasEntities db = new pautasEntities())
   {
      pessoa existe = db.pessoas.Find(nif);
      if (existe == null)
      {
         return JavaScript("alert('Não existe qualquer pessoa com este nif')");
      }
      aluno estealuno = db.alunos.Where(x => x.nif.Equals(nif)).FirstOrDefault();
      if (estealuno == null)
      {
         estealuno = new aluno() { nif = nif, num = -1 };
         ViewBag.existe = false;
      }
      else ViewBag.existe = true;

      return PartialView(estealuno);

   }
}
[HttpPost]
public ActionResult PopUpAluno(aluno a)
{
   using (pautasEntities db = new pautasEntities())
   {

      try
      {

         int num = -1;
         aluno este = db.alunos.Where(x => x.nif == a.nif).SingleOrDefault();
         if (este != null)
         {
            //este.num = a.num;
            este.observacoes = a.observacoes;
            este.dataEntrada = a.dataEntrada;
```

```
                este.matriculaValidaAte = a.matriculaValidaAte;
                db.SaveChanges();
            }
            else
            {
                num = a.num;
                db.alunos.Add(a);
                db.SaveChanges();
            }
            return Json(new { msg = "ok", numero = num }, JsonRequestBehavior.AllowGet);

        }
        catch (Exception erro)
        {

            return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
        }


    }

}
//-----------------------------------------------------------------------------------------------------------------
PopUpFuncionario----------------------

public ActionResult DevolveNum(string nif)
{

    using (pautasEntities db = new pautasEntities())
    {
        aluno este = db.alunos.Where(x => x.nif == nif).FirstOrDefault();
        if (este != null) {
            return Json(new { num=este.num},JsonRequestBehavior.AllowGet);
        } else {
            return Json(new { num = 0 }, JsonRequestBehavior.AllowGet);
        }
```

```
        }
    }

    public ActionResult EditAlunoTurma(int? idturma, int? num)
    {
        using (pautasEntities db = new pautasEntities())
        {

            alunosturma este = db.alunosturmas.Where(a => a.idturma == idturma && a.num == num).FirstOrDefault();
            if (este != null) return PartialView(este);
            // else return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg = "erro" });
            else return new EmptyResult();
        }
    }

    [HttpPost]
    public ActionResult EditAlunoTurma(alunosturma at)
    {
        using (pautasEntities db = new pautasEntities())
        {
            try
            {
                alunosturma este = db.alunosturmas.Where(a => a.idturma == at.idturma && a.num == at.num).FirstOrDefault();

                if (este != null)
                {
                    este.datafim = at.datafim;
                    db.SaveChanges();

                }
```

```
                return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg = "Atualizado
com sucesso" });

            }
            catch (Exception erro)
            {

                return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg =
erro.Message });
            }

        }
    }

    public ActionResult AddAlunoTurma(int? idturma)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                int idt = idturma ?? db.turmas.First().idturma;

                alunosturma novo = new alunosturma() { idturma = idt, datainicio =
DateTime.Now, datafim = null };
                return PartialView("AddAlunoTurma",novo);



            }
        }
        catch (Exception)
        {

            throw;
        }
    }
```

```
        [HttpPost]
        public ActionResult AddAlunoTurma(alunosturma at, string nifs)
        {
          try
          {
            if (ModelState.IsValid)
            {
              using (pautasEntities db = new pautasEntities()) {
                aluno a = db.alunos.Where(x => x.nif == nifs).FirstOrDefault();
                if (a == null)
                {
                  aluno novo = new aluno() {num=at.num,nif=nifs };
                  db.alunos.Add(novo);
                }
                db.alunosturmas.Add(at);

                db.SaveChanges();
                return RedirectToAction("EditarAlunosTurma", new { msg = "Aluno inserido
com sucesso" });
              }

            }
            else
            {
              return View(at);
            }
          }
          catch (Exception erro)
          {

            return RedirectToAction("EditarAlunosTurma",new {msg= erro.Message });
          }

        }

        public ActionResult DeleteTurma(int? id)
```

```
        {
            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    turma morta = db.turmas.Find(id ?? -1);
                    if (morta != null)
                    {
                        db.turmas.Remove(morta);
                        db.SaveChanges();
                        return Json(new { msg = "Turma Eliminada com sucesso" }, JsonRequestBehavior.AllowGet);
                    }
                    else
                    {

                        return Json(new { msg = "Erro na Eliminação da Turma" }, JsonRequestBehavior.AllowGet);
                    }

                }
            }
            catch (Exception erro)
            {
                return Json(new { msg = "Não é possível apagar a turma cod:" + erro.InnerException.HResult.ToString() }, JsonRequestBehavior.AllowGet);

            }

        }


        // GET: Turmas
        public ActionResult ListarTurmas()
        {
```

```
    return View();
}

public ActionResult GetAlunosTurma(int? idt)
{
    using (pautasEntities db = new pautasEntities())
    {
        List<clsVwAluno> alunos;
        int i = idt ?? 0;
        if (i == 0)
        {
            alunos = (from a in db.vw_alunos
                      select new clsVwAluno()
                      {
                          CodTurma = a.codTurma,
                          Num = a.num,
                          Nif = a.nif,
                          Nome = a.nome,
                          Datanasc = a.datanasc,
                          Idade = a.idade,
                          Foto = a.foto
                      }).ToList();
        }
        else
        {
            alunos = (from a in db.vw_alunos
                      where a.idturma == i
                      select new clsVwAluno()
                      {
                          CodTurma = a.codTurma,
                          Num = a.num,
                          Nif = a.nif,
                          Nome = a.nome,
                          Datanasc = a.datanasc,
                          Idade = a.idade,
                          Foto = a.foto
```

```
                }).ToList();

        }
        return Json(new { data = alunos }, JsonRequestBehavior.AllowGet);
    }
}
public ActionResult GetTurmas()
{
    using (pautasEntities db = new pautasEntities())
    {
        List<clsTurmas> turmas = (from t in db.turmas
                                  select new clsTurmas()
                                  {

                                      nome = t.nome,
                                      codCurso = t.codCurso,
                                      anoEntrada = t.anoEntrada,
                                      ano = t.ano,
                                      modalidade = t.modalidade,
                                      UC = t.UC,
                                      codTurma = t.codturma,
                                      idturma = t.idturma

                                  }).ToList();
        return Json(new { data = turmas }, JsonRequestBehavior.AllowGet);
    }
}

[AllowAnonymous]
public ActionResult AddEdit(int? id)
{
    using (pautasEntities db = new pautasEntities())
    {
        List<curso> cursos = db.cursos.ToList();
        ViewBag.cursos = new SelectList(cursos, "codCurso", "curso1");
```

```
        List<clsModalidade> modalidades = new List<clsModalidade>() {
            new clsModalidade(){Modalidade="Laboral"},
            new clsModalidade(){Modalidade="Pós-Laboral"},
            new clsModalidade(){Modalidade="UC-Lab"},
            new clsModalidade(){Modalidade="UC-Pós-Lab"}
        };
        ViewBag.modalidades = new SelectList(modalidades, "Modalidade", "Modalidade");
        turma t;
        if (id != null)
        {
            t = db.turmas.Find(id);
            if (t == null) t = new turma();
        }
        else
        {
            t = new turma();
        }

        return View(t);
    }


}
[AllowAnonymous]
[HttpPost]
public ActionResult AddEdit(turma t)
{
    using (pautasEntities db = new pautasEntities())
    {

        try
        {
            turma esta;
            if (t.idturma != 0)
            {
                esta = db.turmas.Where(x => x.idturma == t.idturma).SingleOrDefault();
```

```
                    if (esta != null)
                    {
                        esta.nome = t.nome;
                        esta.codCurso = t.codCurso;
                        esta.anoEntrada = t.anoEntrada;
                        esta.modalidade = t.modalidade;
                        esta.UC = t.UC;
                        db.SaveChanges();
                    }
                }
                else
                {
                    db.turmas.Add(t);
                    db.SaveChanges();
                }
                return Json(new { msg = "Turma Gravada com sucesso" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception erro)
            {
                return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
            }
        }  }


        public ActionResult EditarAlunosTurma(int? id, String msg)
        {
            using (pautasEntities db = new pautasEntities())
            {
                ViewBag.msg = msg;
                List<turma> turmas = db.turmas.ToList();
                ViewBag.turmas = new SelectList(turmas, "idturma", "codTurma");
                int idt = db.turmas.Min(ti => ti.idturma);
                turma t = db.turmas.Find(id ?? idt);
                return View(t);
            }
```

```
        }

        public ActionResult LibertaAlunoTurma(int num, int idturma)
        {
            using (pautasEntities db = new pautasEntities())
            {
                alunosturma este = db.alunosturmas.Where(x => x.num == num && x.idturma ==
idturma ).FirstOrDefault();
                if (este != null) {
                    db.alunosturmas.Remove(este);
                    db.SaveChanges();
                    return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);        }
                 return Json(new { msg = "Erro" }, JsonRequestBehavior.AllowGet);
            }
        }


        public ActionResult MostraAluno(string nif)
        {
            using (pautasEntities db = new pautasEntities())
            {
                aluno este = db.alunos.Where(a => a.nif == nif).SingleOrDefault();
                if (este != null)
                {


                    return Json(new
                    {
                        msg = "ok",
                        Num = este.num,
                        Nif = este.nif,
                        MatriculaValidaAte = este.matriculaValidaAte ?? new DateTime(),
                        Observacoes = este.observacoes ?? "",
                        DataEntrada = este.dataEntrada ?? new DateTime()
                    }, JsonRequestBehavior.AllowGet);
                }
```

```
else
{
   return Json(new { msg = "erro" }, JsonRequestBehavior.AllowGet);
}          }     }  } }
```

13 ANEXO C <TURMAS CONTROLLER>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using pautas.Models;
using Newtonsoft.Json;

namespace pautas.Controllers
{
   [Authorize(Roles = "Funcionario,Docente")]
   public class TurmasController : Controller
   {
      public ActionResult AlunoTurmaInsert(alunosturma alunoturma)
      {
         try {
            using (pautasEntities db = new pautasEntities())       {
               alunosturma nova = new alunosturma() {
                        num = alunoturma.num,
                        idturma = alunoturma.idturma,
                        datainicio = alunoturma.datainicio,
                        datafim = alunoturma.datafim };
               db.alunosturmas.Add(nova);
               db.SaveChanges();
               var turmasaluno = (from at in db.alunosturmas.Where(x => x.num ==
                                                                  alunoturma.num)
                              join t in db.turmas on at.idturma equals t.idturma
                              select new { num = at.num, idturma = at.idturma, t.codturma,
                               datainicio = at.datainicio, datafim = at.datafim }).ToList();
return Json(new { count = turmasaluno.Count(), turmasAluno = turmasaluno, msg = "ok" },
                                          JsonRequestBehavior.AllowGet);   }          }
```

```
catch (Exception erro)   {
  using (pautasEntities db = new pautasEntities())        {
                    var turmasaluno = (
                          from at in db.alunosturmas.Where(x => x.num ==
                                                                alunoturma.num)
                           join t in db.turmas on at.idturma equals t.idturma
                           select new { num = at.num, idturma = at.idturma, t.codturma, datainicio =
                                                          at.datainicio, datafim = at.datafim }).ToList();
                    return Json(new { count = -1, turmasAluno = turmasaluno, msg = erro.Message },
                          JsonRequestBehavior.AllowGet);        }    }
        public ActionResult AlunoTurmaEdit(alunosturma newAluno, alunosturma oldAluno)   {
            try {
             using (pautasEntities db = new pautasEntities())     {
                    alunosturma velho = new alunosturma() { num = oldAluno.num, idturma =
oldAluno.idturma, datainicio = oldAluno.datainicio, datafim = oldAluno.datafim };
                    alunosturma nova = new alunosturma() { num = newAluno.num, idturma =
newAluno.idturma, datainicio = newAluno.datainicio, datafim = newAluno.datafim };
                    alunosturma este = db.alunosturmas.Where(x => x.num == velho.num &&
x.idturma == velho.idturma && x.datainicio == velho.datainicio).FirstOrDefault();
                    if (este != null)
                    {
                       db.alunosturmas.Remove(este);
                       db.SaveChanges();
                    }
                    db.alunosturmas.Add(nova);
                    db.SaveChanges();
                    var turmasaluno = (from at in db.alunosturmas.Where(x => x.num == nova.num)
                                join t in db.turmas on at.idturma equals t.idturma
                                select new { num = at.num, idturma = at.idturma, t.codturma,
datainicio = at.datainicio, datafim = at.datafim }).ToList();
                    return Json(new { count = turmasaluno.Count(), turmasAluno = turmasaluno, msg
= "Registo Editado com sucesso" }, JsonRequestBehavior.AllowGet);

                  }

                }
                catch (Exception erro)
```

```
            {
                using (pautasEntities db = new pautasEntities())
                {
                    var turmasaluno = (from at in db.alunosturmas.Where(x => x.num == newAluno.num)
                                       join t in db.turmas on at.idturma equals t.idturma
                                       select new { num = at.num, idturma = at.idturma, t.codturma, datainicio = at.datainicio, datafim = at.datafim }).ToList();

                    return Json(new { count = 0, turmasAluno = turmasaluno, msg = erro.Message }, JsonRequestBehavior.AllowGet);
                }
            }
        }
        public ActionResult alunoturmadel(int idturma, int num)
        {

            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    alunosturma item = db.alunosturmas.Where(x => x.idturma == idturma && x.num == num).FirstOrDefault();
                    if (item != null)
                    {
                        db.alunosturmas.Remove(item);
                        db.SaveChanges();
                        return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);
                    }
                }
            }
            catch (Exception e)
            {

                return Json(new { msg = e.Message }, JsonRequestBehavior.AllowGet);
            }
```

```
            return Json(new { msg = "erro" }, JsonRequestBehavior.AllowGet);
        }
public ActionResult PopUpAlunoDel(string nif)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    aluno a = db.alunos.Where(x => x.nif == nif).SingleOrDefault();
                    if (a != null)
                    {
                        db.alunos.Remove(a);
                        pessoa p = db.pessoas.Find(nif);
                        if (p != null) p.estatuto = "";
                        db.SaveChanges();
                        return Json(new { msg = "Aluno Eliminado com sucesso" },
JsonRequestBehavior.AllowGet);
                    }
                    else return Json(new { msg = "Aluno não existe" },
JsonRequestBehavior.AllowGet);
                }
                catch (Exception erro)
                {
                    return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
                }    }
        public ActionResult PopUpAluno(string nif)
        {
            using (pautasEntities db = new pautasEntities())
            {

                pessoa existe = db.pessoas.Find(nif);
                if (existe == null)
                {
                    return JavaScript("alert('Não existe qualquer pessoa com este nif')");
                }
```

```
        aluno estealuno = db.alunos.Where(x => x.nif.Equals(nif)).FirstOrDefault();

        if (estealuno == null)
        {
            estealuno = new aluno() { nif = nif, num = -1 };
            ViewBag.existe = false;
        }
        else ViewBag.existe = true;

        return PartialView(estealuno);

    }

}
[HttpPost]
public ActionResult PopUpAluno(aluno a)
{
    using (pautasEntities db = new pautasEntities())
    {

        try
        {

            int num = -1;
            aluno este = db.alunos.Where(x => x.nif == a.nif).SingleOrDefault();
            if (este != null)
            {
                //este.num = a.num;
                este.observacoes = a.observacoes;
                este.dataEntrada = a.dataEntrada;
                este.matriculaValidaAte = a.matriculaValidaAte;
                db.SaveChanges();
            }
            else
            {
```

```
                num = a.num;
                db.alunos.Add(a);
                db.SaveChanges();
            }
            return Json(new { msg = "ok", numero = num }, JsonRequestBehavior.AllowGet);

        }
        catch (Exception erro)
        {

            return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
        }


    }

}
//-------------------------------------------------------------------------------------------------------------
PopUpFuncionario----------------------

public ActionResult DevolveNum(string nif)
{

    using (pautasEntities db = new pautasEntities())
    {
        aluno este = db.alunos.Where(x => x.nif == nif).FirstOrDefault();
        if (este != null) {
            return Json(new { num=este.num},JsonRequestBehavior.AllowGet);
        } else {
            return Json(new { num = 0 }, JsonRequestBehavior.AllowGet);
        }

    }
}
public ActionResult EditAlunoTurma(int? idturma, int? num) {
    using (pautasEntities db = new pautasEntities())
```

```
        {
         alunosturma este = db.alunosturmas.Where(a => a.idturma == idturma && a.num == num).FirstOrDefault();
                if (este != null) return PartialView(este);
                // else return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg = "erro" });
                else return new EmptyResult();
              }
            }

            [HttpPost]
            public ActionResult EditAlunoTurma(alunosturma at)
            {
              using (pautasEntities db = new pautasEntities())
              {
                try
                {
                  alunosturma este = db.alunosturmas.Where(a => a.idturma == at.idturma && a.num == at.num).FirstOrDefault();

                  if (este != null)
                  {
                    este.datafim = at.datafim;
                    db.SaveChanges();

                  }

                  return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg = "Atualizado com sucesso" });

                }
                catch (Exception erro)
                {
                   return RedirectToAction("EditarAlunosTurma", "Turmas", new { msg =
                    erro.Message });
                }        }      }
```

```
public ActionResult AddAlunoTurma(int? idturma)
{
   try
   {
      using (pautasEntities db = new pautasEntities())
      {
         int idt = idturma ?? db.turmas.First().idturma;
         alunosturma novo = new alunosturma() { idturma = idt, datainicio =
                        DateTime.Now, datafim = null };
         return PartialView("AddAlunoTurma",novo);
      }
   }
   catch (Exception)
   {       throw;
   }
}

[HttpPost]
public ActionResult AddAlunoTurma(alunosturma at, string nifs)       {
   try
   {
      if (ModelState.IsValid)
      {
         using (pautasEntities db = new pautasEntities()) {
            aluno a = db.alunos.Where(x => x.nif == nifs).FirstOrDefault();
            if (a == null)
            {
               aluno novo = new aluno() {num=at.num,nif=nifs };
               db.alunos.Add(novo);
            }
            db.alunosturmas.Add(at);

            db.SaveChanges();
            return RedirectToAction("EditarAlunosTurma", new { msg = "Aluno inserido
com sucesso" });
         }
```

```
            }
            else
            {
                return View(at);
            }
        }
        catch (Exception erro)
        {

            return RedirectToAction("EditarAlunosTurma",new {msg= erro.Message });
        }

    }

    public ActionResult DeleteTurma(int? id)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                turma morta = db.turmas.Find(id ?? -1);
                if (morta != null)
                {
                    db.turmas.Remove(morta);
                    db.SaveChanges();
                    return Json(new { msg = "Turma Eliminada com sucesso" },
JsonRequestBehavior.AllowGet);
                }
                else
                {

                    return Json(new { msg = "Erro na Eliminação da Turma" },
JsonRequestBehavior.AllowGet);
                }
```

```
            }
        }
        catch (Exception erro)
        {
            return Json(new { msg = "Não é possível apagar a turma cod:" +
erro.InnerException.HResult.ToString() }, JsonRequestBehavior.AllowGet);

        }

    }


    // GET: Turmas
    public ActionResult ListarTurmas()
    {

        return View();
    }

    public ActionResult GetAlunosTurma(int? idt)
    {
        using (pautasEntities db = new pautasEntities())
        {
            List<clsVwAluno> alunos;
            int i = idt ?? 0;
            if (i == 0)
            {
                alunos = (from a in db.vw_alunos
                          select new clsVwAluno()
                          {
                              CodTurma = a.codTurma,
                              Num = a.num,
                              Nif = a.nif,
                              Nome = a.nome,
                              Datanasc = a.datanasc,
                              Idade = a.idade,
```

```
                Foto = a.foto
            }).ToList();
        }
        else
        {
            alunos = (from a in db.vw_alunos
                where a.idturma == i
                select new clsVwAluno()
                {
                    CodTurma = a.codTurma,
                    Num = a.num,
                    Nif = a.nif,
                    Nome = a.nome,
                    Datanasc = a.datanasc,
                    Idade = a.idade,
                    Foto = a.foto
                }).ToList();

        }
        return Json(new { data = alunos }, JsonRequestBehavior.AllowGet);
    }
}
public ActionResult GetTurmas()       {
    using (pautasEntities db = new pautasEntities())       {
        List<clsTurmas> turmas = (from t in db.turmas
                                  select new clsTurmas()
                                  {

                                      nome = t.nome,
                                      codCurso = t.codCurso,
                                      anoEntrada = t.anoEntrada,
                                      ano = t.ano,
                                      modalidade = t.modalidade,
                                      UC = t.UC,
                                      codTurma = t.codturma,
                                      idturma = t.idturma
```

```
                        }).ToList();
        return Json(new { data = turmas }, JsonRequestBehavior.AllowGet);
    }
}

[AllowAnonymous]
public ActionResult AddEdit(int? id)
{
    using (pautasEntities db = new pautasEntities())
    {
        List<curso> cursos = db.cursos.ToList();
        ViewBag.cursos = new SelectList(cursos, "codCurso", "curso1");

        List<clsModalidade> modalidades = new List<clsModalidade>() {
            new clsModalidade(){Modalidade="Laboral"},
            new clsModalidade(){Modalidade="Pós-Laboral"},
            new clsModalidade(){Modalidade="UC-Lab"},
            new clsModalidade(){Modalidade="UC-Pós-Lab"}
        };
        ViewBag.modalidades = new SelectList(modalidades, "Modalidade", "Modalidade");
        turma t;
        if (id != null)
        {
            t = db.turmas.Find(id);
            if (t == null) t = new turma();
        }
        else
        {
            t = new turma();
        }

        return View(t);
    }
```

```
        }
        [AllowAnonymous]
        [HttpPost]
        public ActionResult AddEdit(turma t)
        {
          using (pautasEntities db = new pautasEntities())
          {

            try
            {
              turma esta;
              if (t.idturma != 0)
              {
                esta = db.turmas.Where(x => x.idturma == t.idturma).SingleOrDefault();
                if (esta != null)
                {
                  esta.nome = t.nome;
                  esta.codCurso = t.codCurso;
                  esta.anoEntrada = t.anoEntrada;
                  esta.modalidade = t.modalidade;
                  esta.UC = t.UC;
                  db.SaveChanges();
                }
              }
              else
              {
                db.turmas.Add(t);
                db.SaveChanges();
              }
              return Json(new { msg = "Turma Gravada com sucesso" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception erro)
            {

              return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
```

```
            }


        }

    }

    public ActionResult EditarAlunosTurma(int? id, String msg)
    {
        using (pautasEntities db = new pautasEntities())
        {
            ViewBag.msg = msg;
            List<turma> turmas = db.turmas.ToList();
            ViewBag.turmas = new SelectList(turmas, "idturma", "codTurma");
            int idt = db.turmas.Min(ti => ti.idturma);
            turma t = db.turmas.Find(id ?? idt);
            return View(t);
        }
    }

    public ActionResult LibertaAlunoTurma(int num, int idturma)
    {
        using (pautasEntities db = new pautasEntities())
        {
            alunosturma este = db.alunosturmas.Where(x => x.num == num && x.idturma ==
idturma ).FirstOrDefault();
            if (este != null) {
                db.alunosturmas.Remove(este);
                db.SaveChanges();

                return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);
            }

            return Json(new { msg = "Erro" }, JsonRequestBehavior.AllowGet);
```

```
    }
}


public ActionResult MostraAluno(string nif)
{
    using (pautasEntities db = new pautasEntities())
    {
        aluno este = db.alunos.Where(a => a.nif == nif).SingleOrDefault();

        if (este != null)
        {


            return Json(new
            {
                msg = "ok",
                Num = este.num,
                Nif = este.nif,
                MatriculaValidaAte = este.matriculaValidaAte ?? new DateTime(),
                Observacoes = este.observacoes ?? "",
                DataEntrada = este.dataEntrada ?? new DateTime()
            }, JsonRequestBehavior.AllowGet);

        }
        else
        {

            return Json(new { msg = "erro" }, JsonRequestBehavior.AllowGet);
        }
    }
}


}
```

```
}
```

14 ANEXO D - < CURSOSDISCIPLINASCONTROLLER>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using pautas.Models;
using PagedList;
using System.Net;
using System.Data.Entity.Core.Objects;
using System.Data.SqlClient;
using OfficeOpenXml;
using System.IO;
using System.Drawing;
using OfficeOpenXml.Style;

namespace pautas.Controllers
{
   public class CursosDisciplinasController : Controller
   {

      #region cursos
      // GET: CursosDisciplinas
      public ActionResult ListarCursos(String msg, int? page)
      {
         ViewBag.msg = msg;
         using (pautasEntities db = new pautasEntities())
         {
            List<curso> cursos = db.cursos.ToList();
            int pagina = page ?? 1;
```

```
            int pagesize = 10;
            return View(cursos.ToPagedList(pagina, pagesize));

        }
    }

    public ActionResult AddCurso()
    {
        try
        {
            curso novo = new curso();
            return View(novo);
        }
        catch (Exception erro)
        {

            return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = erro.Message });
        }

    }
    [HttpPost]
    public ActionResult AddCurso(curso novo)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                db.cursos.Add(novo);
                db.SaveChanges();
                return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = "Registo Gravado com sucesso" });
            }
        }
        catch (Exception erro)
        {
```

```
                return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = erro.Message });
            }

        }
        public ActionResult EditCurso(string codcurso)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    curso este = db.cursos.Find(codcurso);
                    if (este != null) return View(este);
                    return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = "Curso não existe" });

                }

                catch (Exception erro)
                {

                    return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = erro.Message });
                }
            }

        }
        [HttpPost]
        public ActionResult EditCurso(curso editado)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    curso este = db.cursos.Find(editado.codCurso);
```

```
                    if (este != null)
                    {
                        este.curso1 = editado.curso1;
                        este.nivel = editado.nivel;
                        db.SaveChanges();

                    }
                    return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = "Curso
não existe" });

                }

                catch (Exception erro)
                {

                    return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg =
erro.Message });
                }
            }

        }
        public ActionResult DeleteCurso(String codcurso)
        {
            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    curso este = db.cursos.Find(codcurso);
                    if (este != null)
                    {
                        db.cursos.Remove(este);
                        db.SaveChanges();
                        return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg =
"Registo Eliminado com Sucesso" });
                    }
```

```
                return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = "O Registo não foi eliminado" });
              }
            }
            catch (Exception erro)
            {

              return RedirectToAction("ListarCursos", "CursosDisciplinas", new { msg = erro.Message });
            }



        }
        #endregion

        #region Disciplinas
        public ActionResult ListarDisciplinas(String msg, int? page)
        {
          using (pautasEntities db = new pautasEntities())
          {
            ViewBag.msg = msg;
            List<disciplina> disciplinas = db.disciplinas.ToList();
            int pagina = page ?? 1;
            int pagesize = 10
    ;
            return View(disciplinas.ToPagedList(pagina, pagesize));


          }
        }

        public ActionResult AddDisciplina()
        {
          try
          {
```

```
            disciplina nova = new disciplina();
            return View(nova);
          }
          catch (Exception erro)
          {

            return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
erro.Message });
          }
        }
        [HttpPost]
        public ActionResult AddDisciplina(disciplina nova)
        {
          try
          {
            using (pautasEntities db = new pautasEntities())
            {
              db.disciplinas.Add(nova);
              db.SaveChanges();
              return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
"Disciplina Inserida com sucesso" });

            }

          }
          catch (Exception erro)
          {

            return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
erro.Message });
          }
        }

        public ActionResult DeleteDisciplina(String coddisc)
        {
          using (pautasEntities db = new pautasEntities())
```

```
            {
                disciplina esta = db.disciplinas.Where(x => x.codDisc.Equals(coddisc)).SingleOrDefault();
                if (esta != null)
                {
                    curriculo cur = db.curriculos.Where(z => z.codDisc.Equals(coddisc)).SingleOrDefault();
                    if (cur != null)
                    {
                        return Json(new { msg = "Esta disciplina não pode ser apagada -tem dependências " }, JsonRequestBehavior.AllowGet);
                    }
                    db.disciplinas.Remove(esta);
                    db.SaveChanges();
                    return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);

                }
                else
                {
                    return Json(new { msg = "Registo não existe" }, JsonRequestBehavior.AllowGet);

                }
            }


        }

        public ActionResult EditDisciplina(String coddisc)
        {
            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    disciplina esta = db.disciplinas.Find(coddisc);
                    if (esta != null)
                    {
```

```
                    return View(esta);
                }
                return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
"Disciplina não Existe" });

            }
        }
        catch (Exception)
        {

            throw;
        }
    }

    [HttpPost]
    public ActionResult EditDisciplina(disciplina alterada)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                disciplina esta = db.disciplinas.Find(alterada.codDisc);
                if (esta != null)
                {
                    esta.disciplina1 = alterada.disciplina1;
                    db.SaveChanges();
                    return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
"Disciplina gravada com sucesso" });

                }
                return RedirectToAction("ListarDisciplinas", "CursosDisciplinas", new { msg =
"Disciplina não Existe" });

            }
        }
        catch (Exception)
```

```
            {

                throw;
            }
        }




        #endregion

        #region curriculos

        public ActionResult ListaCurriculos()
        {
            using (pautasEntities db = new pautasEntities())
            {
                List<ClsCurso> cursos = (from c in db.cursos
                                         select new ClsCurso() { CodCurso = c.codCurso, Curso = c.curso1, Nivel
= c.nivel }).ToList();
                List<ClsCurriculo> curriculos = (from cur in db.curriculos
                                                 select new ClsCurriculo
                                                 {
                                                     IdCurr = cur.idcurr,
                                                     CodCurso = cur.codCurso,
                                                     CodDisc = cur.codDisc,
                                                     Ects = cur.ects,
                                                     Ano = cur.ano,
                                                     Semestre = cur.semestre,
                                                     CodUnidade = cur.codUnidade,
                                                     Precedencia = cur.precedencia
                                                 }).ToList();
                List<ClsDisciplina> disciplinas = (from d in db.disciplinas
                                                   select new ClsDisciplina { CodDisc = d.codDisc, Disciplina =
d.disciplina1 }).ToList();
```

```
                return Json(new { Cursos = cursos, Curriculos = curriculos, Disciplinas = disciplinas },
JsonRequestBehavior.AllowGet);


            }

        }
        public ActionResult Curriculos(string msg)
        {
            ViewBag.msg = msg;
            return View();
        }
        [HttpPost]
        public ActionResult GravaCurriculo(ClsCurriculo cur)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    var c = db.curriculos.Find(cur.IdCurr);
                    if (c != null)
                    {
                        c.codCurso = cur.CodCurso;
                        c.codDisc = cur.CodDisc;
                        c.ects = cur.Ects;
                        c.ano = cur.Ano;
                        c.semestre = cur.Semestre;
                        c.codUnidade = cur.CodUnidade;
                        c.precedencia = cur.Precedencia;
                        db.SaveChanges();
                        return Json(new { msg = "Currículo gravado com sucesso" },
JsonRequestBehavior.AllowGet);
                    }
                    else
                    {
```

```
                    return Json(new { msg = "Currículo não Existe" },
JsonRequestBehavior.AllowGet);
                }


            }
            catch (Exception)
            {

                return Json(new { msg = "Erro" }, JsonRequestBehavior.AllowGet);
            }

        }
    }


    public ActionResult DeleteCurriculo(int? id)
    {
        try
        {
            using (pautasEntities db = new pautasEntities())
            {
                curriculo este = db.curriculos.Find(id ?? -1);
                if (este != null)
                {
                    db.curriculos.Remove(este);
                    db.SaveChanges();
                    return Json(new { msg = "ok" }, JsonRequestBehavior.AllowGet);
                }
                else
                {

                    return Json(new { msg = "Registo não Existe" }, JsonRequestBehavior.AllowGet);
                }
            }
        }
```

```
            catch (Exception)
            {

                return Json(new { msg = "Erro ao eliminar registo" },
JsonRequestBehavior.AllowGet);
            }
        }


        public ActionResult AddCurriculo()
        {
            using (pautasEntities db = new pautasEntities())
            {

                curriculo este = new curriculo();
                return PartialView(este);
            }
        }


        [HttpPost]
        public ActionResult AddCurriculo(curriculo novo)
        {
            using (pautasEntities db = new pautasEntities())
            {

                db.curriculos.Add(novo);
                db.SaveChanges();
                return RedirectToAction("Curriculos", "CursosDisciplinas", new { msg = "Currículo
Inserido com sucesso" });
            }
        }
        #endregion

        #region aulas
```

```
        public ActionResult ListarAulas(String msg, string dropseekanosletivos, string
dropseekdocentes, int? dropseekcurriculos, int? dropseekturmas)
        {
          ViewBag.msg = msg;
          ViewBag.dropseekanosletivos = dropseekanosletivos;
          using (pautasEntities db = new pautasEntities())
          {

            List<ClsDocenteItem> lstdocentes = new List<ClsDocenteItem>();
            lstdocentes.Add(new ClsDocenteItem() { Nif = "", Nome = "" });
            lstdocentes.AddRange((from p in db.pessoas
                                  where p.estatuto.Equals("Docente")
                                  orderby p.nome
                                  select new ClsDocenteItem { Nif = p.nif, Nome = p.nome }).ToList());
            ViewBag.docentes = new SelectList(lstdocentes, "Nif", "Nome");

            List<ClsCurriculoitem> lstcurriculos = new List<ClsCurriculoitem>();
            lstcurriculos.Add(new ClsCurriculoitem() { IdCurr = null, Descritivo = "" });
            lstcurriculos.AddRange((from c in db.curriculos
                                    orderby c.idcurr
                                    let desc = c.codCurso + "_" + c.codDisc + "_Ano_" + c.ano.ToString()
                                    select new ClsCurriculoitem { IdCurr = c.idcurr, Descritivo = desc }
                                            ).ToList());

            ViewBag.curriculos = new SelectList(lstcurriculos, "IdCurr", "Descritivo");

            List<ClsTurmaitem> lstturmas = new List<ClsTurmaitem>();
            lstturmas.Add(new ClsTurmaitem { idturma = null, codTurma = "" });
            lstturmas.AddRange((from t in db.turmas
                                select new ClsTurmaitem { idturma = t.idturma, codTurma = t.codturma }
                                        ).ToList());
            ViewBag.turmas = new SelectList(lstturmas, "idturma", "codTurma");
```

```
            List<ClsAula> Aulas = (from a in db.aulas
                                   select new ClsAula
                                   {
                                      Idaula = a.idaula,
                                      Nif = a.nif,
                                      Idcurr = a.idcurr,
                                      Idturma = a.idturma,
                                      Anoletivo = a.anoletivo
                                   }).ToList();


            List<ClsAnoLetivoItem> anosletivos = new List<ClsAnoLetivoItem>();
            anosletivos.Add(new ClsAnoLetivoItem { AnoLetivoValue = "", AnoLetivoText = "" });
            var zeta = db.aulas.GroupBy(X => X.anoletivo).Select(X => X.FirstOrDefault());
            anosletivos.AddRange((from c in zeta
                                  select new ClsAnoLetivoItem { AnoLetivoValue = c.anoletivo,
AnoLetivoText = c.anoletivo }
                                      ).ToList());
            ViewBag.anosletivos = anosletivos;



            if (!String.IsNullOrEmpty(dropseekdocentes)) Aulas = Aulas.Where(a =>
a.Nif.Equals(dropseekdocentes.ToString())).ToList();
            if (dropseekcurriculos != null) Aulas = Aulas.Where(a => a.Idcurr ==
dropseekcurriculos).ToList();
            if (dropseekturmas != null) Aulas = Aulas.Where(a => a.Idturma ==
dropseekturmas).ToList();
            if (!String.IsNullOrEmpty(dropseekanosletivos)) Aulas = Aulas.Where(a =>
a.Anoletivo == dropseekanosletivos).ToList();



            return View(Aulas);
        }
    }
```

```
        //-----------------------------------------------------

        public ActionResult Addaula()
        {
          using (pautasEntities db = new pautasEntities())
          {
            List<ClsDocenteItem> lstdocentes = (from p in db.pessoas
                                  where p.estatuto.Equals("Docente")
                                  orderby p.nome
                                  select new ClsDocenteItem { Nif = p.nif, Nome = p.nome
}).ToList();
            ViewBag.docentes = new SelectList(lstdocentes, "Nif", "Nome");

            List<ClsCurriculoitem> lstcurriculos = (from c in db.curriculos
                                    orderby c.idcurr
                                    let desc = c.codCurso + "_" + c.codDisc + "_Ano_" +
c.ano.ToString()
                                    select new ClsCurriculoitem { IdCurr = c.idcurr, Descritivo =
desc }
                                   ).ToList();

            ViewBag.curriculos = new SelectList(lstcurriculos, "IdCurr", "Descritivo");

            List<ClsTurmaitem> lstturmas = (from t in db.turmas
                                select new ClsTurmaitem { idturma = t.idturma, codTurma =
t.codturma }
                                ).ToList();
            ViewBag.turmas = new SelectList(lstturmas, "idturma", "codTurma");
            aula nova = new aula();

            return View(nova);

          }
        }
```

```
        [HttpPost]
        public ActionResult Addaula(aula nova)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    db.aulas.Add(nova);
                    db.SaveChanges();

                    return RedirectToAction("ListarAulas", new { msg = "Registo Inserido com sucesso." });
                }
                catch (Exception erro)
                {

                    return RedirectToAction("ListarAulas", new { msg = erro.Message });
                }
            }

        }


        public ActionResult Editaula(int? idaula)
        {
            using (pautasEntities db = new pautasEntities())
            {
                List<ClsDocenteItem> lstdocentes = (from p in db.pessoas
                                                    where p.estatuto.Equals("Docente")
                                                    orderby p.nome
                                                    select new ClsDocenteItem { Nif = p.nif, Nome = p.nome }).ToList();
                ViewBag.docentes = new SelectList(lstdocentes, "Nif", "Nome");

                List<ClsCurriculoitem> lstcurriculos = (from c in db.curriculos
                                                        orderby c.idcurr
```

```
                                    let desc = c.codCurso + "_" + c.codDisc + "_Ano_" +
c.ano.ToString()
                                    select new ClsCurriculoitem { IdCurr = c.idcurr, Descritivo =
desc }
                                   ).ToList();

            ViewBag.curriculos = new SelectList(lstcurriculos, "IdCurr", "Descritivo");

            List<ClsTurmaitem> lstturmas = (from t in db.turmas
                                select new ClsTurmaitem { idturma = t.idturma, codTurma =
t.codturma }
                                ).ToList();
            ViewBag.turmas = new SelectList(lstturmas, "idturma", "codTurma");
            aula esta = db.aulas.Find(idaula);
            if (esta != null) return View(esta);
            return RedirectToAction("ListarAulas", new { msg = "Registo não existe" });
          }
        }

        [HttpPost]
        public ActionResult Editaula(aula alterada)
        {
          using (pautasEntities db = new pautasEntities())
          {
            try
            {
              aula esta = db.aulas.Find(alterada.idaula);
              if (esta != null)
              {
                esta.idcurr = alterada.idcurr;
                esta.idturma = alterada.idturma;
                esta.nif = alterada.nif;
                esta.anoletivo = alterada.anoletivo;
                db.SaveChanges();
                return RedirectToAction("ListarAulas", new { msg = "Registo alterado com
sucesso" });
```

```
                }
                return RedirectToAction("ListarAulas", new { msg = "Registo não existe" });
            }
            catch (Exception erro)
            {

                return RedirectToAction("ListarAulas", new { msg = erro.Message });
            }
        }
    }

    public ActionResult DeleteAula(int? idaula)
    {
        using (pautasEntities db = new pautasEntities())
        {
            try
            {
                aula esta = db.aulas.Find(idaula);
                if (esta != null)
                {
                    db.aulas.Remove(esta);
                    db.SaveChanges();
                    return Json(new { msg = "Registo eliminado com sucesso" }, JsonRequestBehavior.AllowGet);
                }
                return Json(new { msg = "Registo não existe" }, JsonRequestBehavior.AllowGet);
            }
            catch (Exception erro)
            {

                return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
            }
        }

    }
```

```
        #endregion


        #region exames
        public ActionResult CreateExame()
        {
           exame novo = new exame() { dataRealizacao = DateTime.Now };
           return View(novo);
        }

        [HttpPost]
        public ActionResult CreateExame(exame novo)
        {
           try
           {
              using (pautasEntities db = new pautasEntities())
              {
                 db.exames.Add(novo);
                 db.SaveChanges();
                 return RedirectToAction("ListarExames", "CursosDisciplinas", new { msg = "Registo
inserido com sucesso" });
              }
           }
           catch (Exception)
           {

              return RedirectToAction("ListarExames", "CursosDisciplinas", new { msg = "Erro na
inserção do registo" });
           }
        }



        [HttpPost]
```

```
        public ActionResult NovaNota(exameNota nova)
        {
          try
          {
            using (pautasEntities db = new pautasEntities())
            {
              db.exameNotas.Add(nova);
              db.SaveChanges();
              return Json(new { msg = "Nota Inserida com sucesso" }, JsonRequestBehavior.AllowGet);
            }
          }
          catch (Exception erro)
          {
            if (erro.InnerException.ToString().Contains("UNIQUE"))
            {
              return Json(new { msg = "O aluno já consta neste exame" }, JsonRequestBehavior.AllowGet);
            }
            return Json(new { msg = erro.InnerException.Message }, JsonRequestBehavior.AllowGet);
          }
        }
        public ActionResult NovaNota(int idexame, int idcurriculo, int idturma)
        {
          try
          {
            using (pautasEntities db = new pautasEntities())
            {

              exameNota nova = new exameNota() { idExame = idexame, idcurr = idcurriculo, idturma = idturma };
              return PartialView(nova);
            }
          }
          catch (Exception erro)
```

```
            {

                throw;
            }

        }
        public ActionResult devolveAlineas()
        {
            using (pautasEntities db = new pautasEntities())
            {
                List<alinea> alineas = db.alineas.ToList();
                alineas.Add(new alinea() { alinea1 = "", descritivo = "" });
                var rslt = from it in alineas
                           select new { alinea = it.alinea1, descritivo = it.descritivo };

                return Json(rslt, JsonRequestBehavior.AllowGet);
            }
        }
        public ActionResult EditaNotaDoExame(int idExameNota, int? nota, String alinea)
        {
            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    exameNota ex = db.exameNotas.Find(idExameNota);
                    if (ex != null)
                    {
                        ex.nota = nota;
                        ex.alinea = alinea;
                        db.SaveChanges();
                        return Json(new { msg = "Registo Editado com sucesso" }, JsonRequestBehavior.AllowGet);
                    }
                    return Json(new { msg = "Erro na Edição do Registo" }, JsonRequestBehavior.AllowGet);
                }
```

```
            }
            catch (Exception erro)
            {

                return Json(new { msg = "Erro : Nota Inválida" }, JsonRequestBehavior.AllowGet);
            }
           }
        public ActionResult ApagaNotaDoExame(int idExame, int IdExameNota)
        {
            try
            {
                using (pautasEntities db = new pautasEntities())
                {
                    exameNota esta = db.exameNotas.Where(x => x.idExame == idExame && x.idExameNotas == IdExameNota).FirstOrDefault();
                    if (esta != null)
                    {
                        db.exameNotas.Remove(esta);
                        db.SaveChanges();
                        if (db.exameNotas.Where(x => x.idExame == idExame).Count() == 0) return Json(new { msg = "Registo Removido", last = true }, JsonRequestBehavior.AllowGet);
                        return Json(new { msg = "Registo Removido", last = false }, JsonRequestBehavior.AllowGet);
                    }
                    return Json(new { msg = "Erro na Remoção do Registo", last = false }, JsonRequestBehavior.AllowGet);
                }


            }
            catch (Exception erro)
            {

                return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
            }
        }
```

```
        public ActionResult DeleteNotasExame(int idExame)
        {
          try
          {
            using (pautasEntities db = new pautasEntities())
            {

              SqlParameter param1 = new SqlParameter("@idexame",
System.Data.SqlDbType.Int);
              param1.Value = idExame;

              SqlParameter param2 = new SqlParameter("@total", System.Data.SqlDbType.Int);
              param2.Direction = System.Data.ParameterDirection.Output;
              db.Database.ExecuteSqlCommand("exec sp_del_notasExame @idexame, @total
out", param1, param2);
              int total = Convert.ToInt32(param2.Value);
              return Json(new { msg = total }, JsonRequestBehavior.AllowGet);

            }

          }
          catch (Exception erro)
          {

            return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
          }
        }
        public ActionResult InsereNotasExames(int idExame)
        {
          try
          {
            using (pautasEntities db = new pautasEntities())
            {

              SqlParameter param1 = new SqlParameter("@idexame",
System.Data.SqlDbType.Int);
```

```
                param1.Value = idExame;
                SqlParameter param2 = new SqlParameter("@total", System.Data.SqlDbType.Int);
                param2.Direction = System.Data.ParameterDirection.Output;
                db.Database.ExecuteSqlCommand("exec sp_notasExame @idexame, @total out",
param1, param2);
                int total = Convert.ToInt32(param2.Value);
                return Json(new { msg = total }, JsonRequestBehavior.AllowGet);
              }

            }
            catch (Exception erro)
            {

              return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);
            }
          }
          public ActionResult GetNotas(int idexame)
          {
            try
            {
              using (pautasEntities db = new pautasEntities())
              {
                List<clsExameNotas> notas = (from n in db.exameNotas
                                             join a in db.alunos on n.num equals a.num
                                             join p in db.pessoas on a.nif equals p.nif
                                             where n.idExame == idexame
                                             select new clsExameNotas()
                                             {
                                               idExameNotas = n.idExameNotas,
                                               idExame = n.idExame,
                                               num = n.num,
                                               nome = p.nome,
                                               nota = n.nota,
                                               extenso = n.extenso,
                                               alinea = n.alinea,
                                               idturma = n.idturma,
```

```
                                    idcurr = n.idcurr
                                }).ToList()
                            ;
                    return Json(new { data = notas }, JsonRequestBehavior.AllowGet);
                }

            }
            catch (Exception)
            {

                throw;
            }
        }
        public ActionResult ListarExames(String msg, string epoca, string nif, string coddisc, int?
idturma, int? ano, int? idexame, int? page)
        {
            ViewBag.msg = msg;
            List<v_exames> exames;
            using (pautasEntities db = new pautasEntities())
            {
                if (idturma == -1) idturma = null;
                if (coddisc == "") coddisc = null;
                if (nif == "") nif = null;
                if (epoca == "") epoca = null;
                if (idexame == -1) idexame = null;
                exames = db.sp_seek_exames(nif, idturma, coddisc, null, epoca, ano,
idexame).ToList<v_exames>();
                int pagesize = 5;
                int pagina = page ?? 1;
                return View(exames.ToPagedList(pagina, pagesize));

            }

        }
        public ActionResult DeleteExame(int? id)
        {
```

```
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    exame este = db.exames.Find(id ?? -1);
                    if (este != null)
                    {
                        db.exames.Remove(este);
                        db.SaveChanges();
                        return Json(new { msg = "Registo Eliminado com sucesso" }, JsonRequestBehavior.AllowGet);
                    }
                    else
                    {
                        return Json(new { msg = "Registo não encontrado" }, JsonRequestBehavior.AllowGet);

                    }
                }
                catch (Exception erro)
                {

                    return Json(new { msg = erro.Message }, JsonRequestBehavior.AllowGet);

                }

            }
        }
        public ActionResult VerExame(int? id)
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    v_exames este = db.v_exames.Where(x => x.idexame == id).FirstOrDefault();
                    if (este != null)
```

```
                {
                    return View(este);
                }
                else return new HttpStatusCodeResult(HttpStatusCode.BadRequest, "Exame não
existe");
            }
            catch (Exception erro)
            {

                return new HttpStatusCodeResult(HttpStatusCode.BadRequest, erro.Message);
            }
        }
    }
    public ActionResult VerExameNotas(int? idexame)
    {
        using (pautasEntities db = new pautasEntities())
        {

            try
            {
                List<clsExameNotas> notas = (from n in db.exameNotas
                                             join a in db.alunos on n.num equals a.num
                                             join p in db.pessoas on a.nif equals p.nif
                                             where n.idExame == idexame
                                             select new clsExameNotas()
                                             {
                                                 idExameNotas = n.idExameNotas,
                                                 idExame = n.idExame,
                                                 num = n.num,
                                                 nome = p.nome,
                                                 nota = n.nota,
                                                 extenso = n.extenso,
                                                 alinea = n.alinea,
                                                 idturma = n.idturma,
                                                 idcurr = n.idcurr
                                             }).ToList();
```

```
                if (notas.Count > 0)
                {
                    return PartialView(notas);
                }
                else {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest,"Notas Zero");
                }

            }
            catch (Exception erro)
            {

                return new HttpStatusCodeResult(HttpStatusCode.BadRequest, erro.Message);
            }
        }
    }

public ActionResult EditExame(int? id)    {
            using (pautasEntities db = new pautasEntities())     {
                try     {
                    TempData["temnotas"] = db.exameNotas.Where(x => x.idExame == id).Count() >
0;
                    v_exames este = db.v_exames.Where(x => x.idexame == id).FirstOrDefault();
                    if (este != null)
                    {
                        return View(este);
                    }
                    else return new HttpStatusCodeResult(HttpStatusCode.BadRequest, "Exame não
existe");
                }
                catch (Exception erro)     {
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest, erro.Message);
                }       }     }

        [HttpPost]
        public ActionResult EditExame(v_exames e)
```

```
        {
            using (pautasEntities db = new pautasEntities())
            {
                try
                {
                    exame este = db.exames.Where(x => x.idexame == e.idexame).FirstOrDefault();
                    if (este != null)
                    {
                        este.idaula = e.idaula;
                        este.epoca = e.epoca;
                        este.homologado = e.homologado;
                    }
                    db.SaveChanges();
                    ViewBag.msg = "Registo gravado com sucesso";
                    v_exames veste = db.v_exames.Where(x => x.idexame ==
este.idexame).FirstOrDefault();
                    return View(veste);
                }
                catch (Exception erro)
                {



                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest, erro.Message);
                }

            }
        }

        #endregion

        public async System.Threading.Tasks.Task<ActionResult> Pauta(int id)
        {
            using (pautasEntities db = new pautasEntities())
            {
                var exame = db.v_exames.Where(x => x.idexame == id).FirstOrDefault();
```

```
String nome = exame.codExame;
ExcelPackage.LicenseContext = LicenseContext.NonCommercial;
string path = Server.MapPath("~/pautasExcel/") + @"\" + nome + ".xlsx";
if (System.IO.File.Exists(path)) System.IO.File.Delete(path);
ExcelPackage pack = new ExcelPackage(path);
var ws = pack.Workbook.Worksheets.Add("Pauta");

var range = ws.Cells["B12:B17"];
range.Style.Font.Bold = true;

ws.Cells["B11"].Value = "Curso:";
ws.Cells["C11"].Value = exame.codCurso;

ws.Cells["B12"].Value = "Ano Lectivo";
ws.Cells["C12"].Value = exame.anolectivo;

ws.Cells["B13"].Value = "Período Lectivo";
ws.Cells["C13"].Value = exame.semestre;

ws.Cells["B14"].Value = "Tipo de Inscrição";
ws.Cells["C14"].Value = exame.epoca;

ws.Cells["B15"].Value = "Disciplina";
ws.Cells["C15"].Value = exame.codDisc + ",";

ws.Cells["B16"].Value = "Turma";
ws.Cells["C16"].Value = exame.codTurma;

ws.Cells["B17"].Value = "Data";
ws.Cells["C17"].Value = exame.dataRealizacao;
ws.Cells["C17"].Style.Numberformat.Format = "dd-MM-yyyy HH:mm";
var notitas = from n in db.v_notasExames.Where(y => y.idExame == id)
        orderby n.num ascending
        let al = n.alinea
        select new { Número = n.num, Nome = n.nome, Nota = n.nota, Extenso = al
== " " || al == null ? n.extenso : n.extenso + " (" + al };
```

```
var alineasUsadas = from n in db.v_notasExames.Where(y => y.idExame == id)
                    join a in db.alineas on n.alinea equals a.alinea1
                    where (n.alinea != null && n.alinea != "")
                    select new AlineasUsadas { alinea = a.alinea1, Descritivo = a.descritivo };


range = ws.Cells["B19:E19"];
ws.Cells["B19"].Value = "Nº do Aluno";
ws.Cells["C19"].Value = "Nome";
ws.Cells["D19"].Value = "Nota";
ws.Cells["E19"].Value = "Extenso";
range.Style.Font.Bold = true;
range.Style.Fill.PatternType = ExcelFillStyle.Solid;
range.Style.Fill.BackgroundColor.SetColor(Color.LightGray);
range.Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

ws.Cells["B20"].LoadFromCollection(notitas.ToList(), false);
range = ws.Cells["D20:D44"];
foreach (var c in range)
{
   int num;
   if (! String.IsNullOrEmpty(c.Value.ToString())) {

      if (int.TryParse(c.Value.ToString(), out num))
      {
         c.Value = num;
      }
      else c.Value = "---";

   } else c.Value = "---";

}
```

```
            int total = notitas.Count();
            int limite = (total > 5) ? total + 19 : 24;
            String area = "B19:E" + limite.ToString();
            range = ws.Cells[area];
            // Assign borders
            range.Style.Border.Top.Style = ExcelBorderStyle.Thin;
            range.Style.Border.Left.Style = ExcelBorderStyle.Thin;
            range.Style.Border.Right.Style = ExcelBorderStyle.Thin;
            range.Style.Border.Bottom.Style = ExcelBorderStyle.Thin;

            ws.Cells[limite + 2, 2].Value = "Nº. de Elementos: " + total.ToString();

            ws.Cells[limite + 5, 2].Value = "Observações:";
            ws.Cells[limite + 5, 2, limite + 5, 3].Merge = true;
            ws.Cells[limite + 5, 2, limite + 5, 3].Style.HorizontalAlignment =
ExcelHorizontalAlignment.Left;

            System.Text.StringBuilder sb = new System.Text.StringBuilder();
            sb.Append("Alíneas: ");
            foreach (var it in alineasUsadas)
            {
               sb.Append(it.alinea + it.Descritivo + "     ");
            }
            sb.Append("; ");
            ws.Cells[limite + 6, 2].Value = sb.ToString();
            ws.Cells[limite + 6, 2, limite + 6, 5].Merge = true;
            ws.Cells[limite + 6, 2, limite + 6, 5].Style.HorizontalAlignment =
ExcelHorizontalAlignment.Left;


            System.Drawing.Image image =
System.Drawing.Image.FromFile(Server.MapPath("~/images/rubrica.png"));
            var excelImage = ws.Drawings.AddPicture("rub", image);
            excelImage.SetPosition(limite + 11, 0, 2, 0);

            image = System.Drawing.Image.FromFile(Server.MapPath("~/images/texto.png"));
```

```
                excelImage = ws.Drawings.AddPicture("text", image);
                excelImage.SetPosition(limite + 6, 0, 1, 0);
                ws.Cells["A1:G100"].AutoFitColumns();
                ws.Column(4).Width = 5;

                ws.Column(3).Width = 50;
                ws.PrinterSettings.PrintArea = ws.Cells["A1:f50"];
                await pack.SaveAsync();
                byte[] bits = pack.GetAsByteArray();
                // return File(bits, "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
                return File(bits, "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet", nome + ".xlsx");
                // FileStream fs = new FileStream(path, FileMode.Open);
                //return new FileStreamResult(fs, "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
            }


        }

    }//class

}//namespace
```

15 ANEXO E - <SEGURANCACONTROLLER>

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using System.Data.SqlClient;
using System.Web.Mvc;
using pautas.Models;

namespace pautas.Controllers
{
   public class SegurancaController : Controller    {
      String cnnstr = System.Configuration.ConfigurationManager
                .ConnectionStrings["DefaultConnection"].ToString();

      public ActionResult UserRoles(String msg)
      {
         ViewBag.msg = msg;
         try
         {
            using (pautasEntities db = new pautasEntities())
            {

               List<AspNetUser> utilizadores = db.AspNetUsers.ToList();
               return View(utilizadores);
            }
         }
         catch (Exception)
         {
```

```
        throw;          }       }


public ActionResult EditUser(String id)     {
   if (id != null)        {
      SqlConnection cnn = new SqlConnection(cnnstr);
      SqlParameter param = new SqlParameter("@UserId", SqlDbType.NVarChar, 128);
      param.Value = id;
      SqlCommand cmd = new SqlCommand("Select * from AspNetUserRoles where
                        UserId = @UserId;");
      cmd.Connection = cnn;
      cmd.Parameters.Add(param);
      SqlDataAdapter adpt = new SqlDataAdapter(cmd);
      SqlCommandBuilder builder = new SqlCommandBuilder(adpt);
      DataTable tab = new DataTable();
      adpt.Fill(tab);
      List<ClsAspNetUserRoles> aspnetuserroles = new List<ClsAspNetUserRoles>();
      foreach (DataRow r in tab.Rows)
      {
         aspnetuserroles.Add(new ClsAspNetUserRoles { RoleId = r[1].ToString(), UserId =
                                    r[0].ToString(), Nif = r[2].ToString() });
      }
      ViewBag.aspnetUserRoles = aspnetuserroles;
   }
   using (pautasEntities db = new pautasEntities())
   {
      List<pessoa> pessoas = new List<pessoa>();
      pessoas.Add(new pessoa { nif = "", nome = "" });
      pessoas.AddRange(db.pessoas.ToList());
      SelectList listpessoas = new SelectList(pessoas, "nif", "nome");
      ViewBag.listpessoas = listpessoas;
      AspNetUser u = db.AspNetUsers.Where(a => a.Id == id).FirstOrDefault();
      List<AspNetRole> roles = db.AspNetRoles.ToList();
      ViewBag.roles = roles;
      return View(u);
```

```
        }
    }

    [HttpPost]
    public ActionResult EditUser(AspNetUser user, String[] grupo)
    {
        using (pautasEntities db = new pautasEntities())
        {
            try
            {
                AspNetUser u = db.AspNetUsers.Where(a => a.Id == user.Id).FirstOrDefault();
                if (u != null)
                {
                    u.Email = u.Email;
                    u.PhoneNumber = user.PhoneNumber;
                    u.UserName = user.UserName;
                    u.Nif = user.Nif;
                    db.SaveChanges();
                    SqlConnection cnn = new SqlConnection(cnnstr);
                    SqlParameter param = new SqlParameter("@UserId", SqlDbType.NVarChar,
128);
                    param.Value = user.Id;
                    SqlCommand cmd = new SqlCommand("Select * from AspNetUserRoles where
UserId = @UserId;");
                    cmd.Connection = cnn;
                    cmd.Parameters.Add(param);
                    SqlDataAdapter adpt = new SqlDataAdapter(cmd);
                    SqlCommandBuilder builder = new SqlCommandBuilder(adpt);
                    DataTable tab = new DataTable();
                    adpt.Fill(tab);
                    foreach (DataRow r in tab.Rows)
                    {
                        r.Delete();
                    }
                    DataRow novarow;
                    if(grupo != null) {
```

```
                    foreach (var r in grupo)
                    {
                       novarow = tab.NewRow();
                       novarow[0] = user.Id;
                       novarow[1] = r;
                       tab.Rows.Add(novarow);
                    }
                       adpt.Update(tab);
                    }


                  }

                   return RedirectToAction("UserRoles", "Seguranca");
                }
                catch (Exception erro)
                {

                   throw;            }         }      }


        public ActionResult DeleteUser(String id)
            {
               try
               {
                  using (pautasEntities db = new pautasEntities()) {

                     AspNetUser user = db.AspNetUsers.Find(id);
                     if (user != null) db.AspNetUsers.Remove(user);
                     db.SaveChanges();
                  }
                  return        Json(new    {    msg    =    "User    eliminado    com    sucesso"    },
JsonRequestBehavior.AllowGet);
               }
               catch (Exception)       {
                  throw;          }
```

```
} } }
```

16 ANEXO F -LÓGICA DO CLIENTE < VIEW CURRÍCULOS>

```
@section curriculos{
    <script>
         var disciplinas,cursos,valores ;
         var tabcurriculos;
         var pendente = false;
         $(function () {
             $("#frm").validate();
             reloadTab();


         });


        function btdelete(id) {
            return "<button class='btn btn-danger' name ='btdelete' data-valor='" +
id + "' type='button' onclick='apaga(this);'>Delete</button>"
        };


         function btedit(id) {
             return "<button class='btn btn-success' name ='btedit' id='" + id +"'
type='button' onclick='edita();'>Edit</button>"
         };

         function btUpdate(id) {
             return "<button class='btn btn-dark' name ='btupdate' data-valor='" +
id + "' type='button' style='margin-top:5px;'
onclick='grava(this);'>Update</button>"
         };


         function btCancel() {
             return "<button class='btn btn-primary' name ='btcancel'  type='button'
onclick='cancela();' style='margin-top:5px;'>Cancel</button>"
         };


         function edita(evt) {
             evt = evt ? evt : window.event;
             evt.preventDefault();
             if (pendente) {
                 alert("Existe um update pendente");
                 return;
```

```
            }
            pendente = true;
            var linha = $(evt.target).closest('tr');
            linha.css('background-color', 'yellowgreen');
            valores = {
                IdCurr: linha.find('td:eq(0)').text(),
                CodCurso: linha.find('td:eq(1)').text(),
                CodDisc: linha.find('td:eq(2)').text(),
                Ects: linha.find('td:eq(3)').text(),
                Ano: linha.find('td:eq(4)').text(),
                Semestre: linha.find('td:eq(5)').text(),
                CodUnidade : linha.find('td:eq(6)').text(),
                Precedencia : linha.find('td:eq(7)').text()
            };
            var dropcursos = $("<select class='form-control' style='width:80px;'
name='dropCur'></select>");
            var opcoescursos = $.map(cursos, function (v, i, a) {
               if (v.CodCurso == valores.CodCurso) {
                    return "<option value='" + v.CodCurso + "' selected='selected'
>" + v.CodCurso + "</option>";
                } else {
                    return "<option value='" + v.CodCurso + "'>" + v.CodCurso +
"</option>";
                }

            });
            dropcursos.append(opcoescursos);
            linha.find('td:eq(1)').html("").append(dropcursos);
            //-------------------------------------
            var dropdisc = $("<select class='form-control' style='width:80px;'
name='dropDisc'></select>");
            var opcoes = $.map(disciplinas, function (v, i, a) {
               if (v.CodDisc==valores.CodDisc) {
                    return "<option value='" + v.CodDisc + "' selected='selected'
>" + v.CodDisc + "</option>";
                } else {
                    return "<option value='" + v.CodDisc + "'>" + v.CodDisc +
"</option>";
                }

            });
            dropdisc.append(opcoes);
            linha.find('td:eq(2)').html("").append(dropdisc);
            linha.find('td:eq(3)').html("<input type='text' name='txtEcts'
style='width:60px;' class='form-control' value=" + valores.Ects + "  />");

            linha.find('td:eq(4)').html("<input type='text' name='txtAno'
style='width:60px;' class='form-control'  value=" + valores.Ano + "  />");
            $("[name='txtAno']").rules("add", {
                required:true,
                max: 3,
                min: 1,
                messages: {
```

```
                    required: "Required input",
                    min: "Mínimo 1º Ano",
                    max: "Máximo 3º Ano"
                }

            });
            linha.find('td:eq(5)').html("<input type='text' name='txtSemestre'
style='width:60px;' class='form-control'  value=" + valores.Semestre + "  />");
            $("[name='txtSemestre']").rules("add", {
                required: true,
                max: 2,
                min: 1,
                messages: {
                    required: "Required input",
                    min: "Mínimo 1º Semestre",
                    max: "Máximo 2º Semestre"
                }

            });

            linha.find('td:eq(6)').html("<input type='text' name='txtCodUnidade'
style='width:120px;' class='form-control'  value=" + valores.CodUnidade + "  />");
            linha.find('td:eq(7)').html("<input type='text' name='txtPrecedencia'
style='width:60px;' class='form-control'  value='" + valores.Precedencia + "'  />");
            linha.find('td:eq(8)').html(btUpdate(evt.target.id) + " " +
btCancel());

         }

         function cancela() {
            pendente = false;
            location.reload();
        }

        function apaga(x) {
            var cod = $(x).data("valor");
            if (!confirm("Quer mesmo apagar o registo?")) {
                return false;
            }

            $.ajax({
                url: '@Url.Action("DeleteCurriculo","CursosDisciplinas")',
                type: 'get',
                dataType: 'json',
                data: { id: cod },
                success: function (data) {
                    if (data.msg == 'ok') {
                        $("h3").html("Registo Eliminado com Sucesso");
                        $(x).closest("tr").remove();
                    } else {
                        $("h3").html(data.msg);
                    }
                },
```

```
                error: function (erro) {
                    $("h3").html(erro);
                }

            });

        }

         function grava(x) {
             try{
                 var linha = $(x).closest('tr');
                 valores = {
                     IdCurr: linha.find('td:eq(0)').text(),
                     CodCurso:
linha.find('td:eq(1)').find("[name='dropCur']").val(),
                     CodDisc:
linha.find('td:eq(2)').find("[name='dropDisc']").val(),
                     Ects: linha.find('td:eq(3)').find("[name='txtEcts']").val(),
                     Ano: linha.find('td:eq(4)').find("[name='txtAno']").val(),
                     Semestre:
linha.find('td:eq(5)').find("[name='txtSemestre']").val(),
                     CodUnidade:
linha.find('td:eq(6)').find("[name='txtCodUnidade']").val(),
                     Precedencia:
linha.find('td:eq(7)').find("[name='txtPrecedencia']").val()
                 };
                 console.log(valores);
                 linha.css('background-color', '#c4c4c4');
                 $.ajax({
                     url: '@Url.Action("GravaCurriculo","CursosDisciplinas")',
                     type: 'post',
                     dataType: 'json',
                     data: { cur: valores },
                     success: function (data) {
                         if (data.msg == "Currículo gravado com sucesso") {
                             location = "/CursosDisciplinas/Curriculos?msg=" +
data.msg;
                         } else {
                             $("h3").html(data.msg);
                         }
                     },
                     error: function (erro) {
                         alert(erro);
                     }

                 });
             }catch(erro){
                 alert(erro);
             }


         }
```

```
        function reloadTab() {
            $.ajax({
                url: '@Url.Action("ListaCurriculos","CursosDisciplinas")',
                type: 'get',
                contentType: "application/json; charset=utf-8",
                dataType:'json',
                data: {},
                success: function (dados) {
                    disciplinas = dados.Disciplinas;
                    cursos = dados.Cursos;
                    console.log(disciplinas);
                    tabcurriculos = $("#tabcur").DataTable({
                        data: dados.Curriculos,
                        columns: [
                            { data: "IdCurr" },
                            { data: "CodCurso" },
                            { data: "CodDisc" },
                            { data: "Ects" },
                            { data: "Ano" },
                            { data: "Semestre" },
                            { data: "CodUnidade" },
                            { data: "Precedencia" },
                            {
                                data: "IdCurr",
                                render: function (data, type, row, meta) {
                                    return btedit(row.IdCurr) + " " +
                                                                    btdelete(row.IdCurr);
                                },
                                width: "200px"

                            }
                        ]
                    });

                },
                error: function () {
                }

            });


        }//reloadtab
    </script>
}
```

17 ANEXO G - LÓGICA DO CLIENTE <VIEW EDIT EXAME >

```
@section exameNotas{
    <script>

        function criarNovaNota(evt) {
            evt = evt ? evt : window.event;
            evt.preventDefault();
            try {

                var nova = {
                    idExame: $("#idExame").val(),
                    num: $("#num").val(),
                    nota: $("#nota").val(),
                    alinea: $("#alinea").val(),
                    idturma: $("#idturma").val(),
                    idcurr: $("#idcurr").val()
                };
                console.log(nova);
                $.ajax({
                    url: '@Url.Action("NovaNota", "CursosDisciplinas")',
                    method: 'post',
                    data: { nova: nova },
                    dataType: 'json',
                    success: function (dados) {
                        console.log(dados);
                        $("#msg").html(dados.msg);
                        $("#modalnota").modal("hide");
                        tabela.ajax.reload();
                    },
                    error: function (erro) {
                        console.log(erro);
                        $("#msg").html("Erro no Servidor Cliente");
                    }

                });

            } catch (e) {
                alert("Erro");
                console.log(e);
            }
        }


        function novoregisto(evt) {
```

```
            evt = evt ? evt : window.event;
            evt.preventDefault();
            $("#tit").html("Novo Registo");
           // alert(@Model.idcurr);
            var url = "/CursosDisciplinas/NovaNota?idexame=" + @Model.idexame +
"&idcurriculo=" + @Model.idcurr + "&idturma="+ @Model.idturma;
            $("#modalnotabody").load(url, function () {
                $("#modalnota").modal("show");
            });

        }

        var tabela;
        function btdel(evt) {
            evt = evt ? evt : window.event;
            if (!confirm("Quer apagar?")) return false;
            $.ajax({
                  url:'@Url.Action("ApagaNotaDoExame", "CursosDisciplinas")',
                  method: 'post',
                  dataType: 'json',
                  data: { idExame:@Model.idexame, IdExameNota:
$(evt.target).data("idnota")},
                  success: function (dados) {
                      tabela.ajax.reload();
                      $("#msg").html(dados.msg);
                      if (dados.last) {
                          $("#btdeletenotas").prop("disabled", true);
                          $("#btinsertnotas").prop("disabled", false);
                      } else {
                          $("#btdeletenotas").prop("disabled", false);
                          $("#btinsertnotas").prop("disabled", true);
                      }
                  },
                  error: function (err) {
                      alert("Erro");
                  }

            });

        }
            //-----------------------------------------------------------------------
--btedit
        var linha;

        function btcancel(evt) {
            evt = evt ? evt : window.event;
            linhaedit = $(evt.target).closest("tr");
            linhaedit.replaceWith(linha);
            $("#msg").html("");
            $("[name='editanota']").attr("disabled", false);
            $("[name='delnota']").attr("disabled", false);
        }
        function btupdate(evt){
```

```
            evt = evt ? evt : window.event;
           linhaedit = $(evt.target).closest("tr");
            $.ajax({
                url: '@Url.Action("EditaNotaDoExame", "CursosDisciplinas")',
                method: 'post',
                dataType: 'json',
                data: { idExameNota: $(evt.target).data('idexamenota'), nota:
linhaedit.find("td:eq(4) #txtnota").val(), alinea: linhaedit.find("td:eq(6)
#selalinea").val() },
                success: function (dados) {
                    console.log(dados);
                    $("#msg").html(dados.msg);
                    tabela.ajax.reload();
                },
                error: function () { }

            });

        }

        function btedit(evt) {
            evt = evt ? evt : window.event;
                linha = $(evt.target).closest("tr");
                var linhaedit = "<tr style='background-color:lightgreen'>";
                linhaedit += "<td>" + linha.find("td:eq(0)").text() +
"</td>";//idExameNotas
                linhaedit += "<td>" + linha.find("td:eq(1)").text() +
"</td>";//idExame
                linhaedit += "<td>" + linha.find("td:eq(2)").text() + "</td>";//num
                linhaedit += "<td>" + linha.find("td:eq(3)").text() + "</td>";//nome
                linhaedit += "<td><input type='text' style='width:60px' id='txtnota'
value='" + linha.find("td:eq(4)").text() + "'/></td>";//nota
                   linhaedit += "<td>" + linha.find("td:eq(5)").text() +
"</td>";//extenso
                   linhaedit += "<td style='width:200px'><select  id='selalinea'
name='selalinea' class='form-control'/></td>";//alinea
                   linhaedit += "<td>" + linha.find("td:eq(7)").text() +
"</td>";//idturma
                   linhaedit += "<td>" + linha.find("td:eq(8)").text() +
"</td>";//idcurr
                   linhaedit += "<td><a class='btn btn-primary' data-idexame='" +
linha.find("td:eq(1)").text()+"' data-idexamenota='" + linha.find("td:eq(0)").text()
+"' onclick='btupdate();'>Upd</a> ";
                   linhaedit += "<a class='btn btn-warning' id='btcancelnovo'
onclick='btcancel();' >Cnc</a></td>";
                   linhaedit += "</tr>";
                linha.replaceWith(linhaedit);
               $.ajax({
                url:'@Url.Action("devolveAlineas", "CursosDisciplinas")',
                method: 'post',
                dataType: 'json',
                success: function (data) {
                    console.log(data);
```

```
                    $.each(data, function (i, item) {
                        $("[name='selalinea']").append($('<option>', {
                            value: item.alinea,
                            text: item.descritivo
                        }));

                    });
                    $("[name='selalinea']").val(linha.find("td:eq(4)").text());
                    $("#msg").text("Edição do Registo");
                    $("[name='editanota']").attr("disabled", true);
                    $("[name='delnota']").attr("disabled", true);

                },
                error: function (err) {
                    alert("Erro");
                }
            });


            }
            //---------------------------------------------------------------------


        $(function () {

            tabela = $("#tblnotas").DataTable({
                "autoWidth": true,
                ajax: {
                    url:'@Url.Action("GetNotas", "CursosDisciplinas")',
                    method: 'GET',
                    data: {idexame:@Model.idexame},
                    datatype: 'json'

                },
                columns: [
                    { data: "idExameNotas" },
                    { data: "idExame" },
                    { data: "num" },
                    { data: "nome" },
                    { data: "nota" },
                    { data: "extenso" },
                    { data: "alinea" },
                    { data: "idturma" },
                    { data: "idcurr" },
                    {
                        data:"idExame",
                        render: function (data, type, row, meta) {
                            return "<input type='button' value='edit'
onclick='btedit();' name='editanota'   data-id='" + row.idExame + "'  class='btn
btn-success'/>  " +
                                "<input type='button' onclick='btdel()' value='del'
name='delnota' data-idnota='" + row.idExameNotas +"'  data-idExame='" + row.idExame
+ "'  class='btn btn-danger'/>";
```

```
                        }
                    }
                ]

                });

            //datatable
            $("#btinsertnotas").click(function (evt) {
                evt = evt ? evt : window.event;
                $.ajax({
                    url:'@Url.Action("InsereNotasExames", "CursosDisciplinas")',
                    method: 'post',
                    dataType: 'json',
                    data: { idExame:@Model.idexame},
                    success: function (dados) {
                        tabela.ajax.reload();
                        $(evt.target).prop("disabled", true);
                        $("#btdeletenotas").prop("disabled", false);
                        if (parseInt(dados.msg) > 0) $("#msg").html("Foram inseridos "
+ dados.msg + " registos");
                        else $("#msg").html("A turma não tem alunos");
                    },
                    error: function (err) {
                        alert("Erro");
                    }

                });

            });

            $("#btdeletenotas").click(function (evt) {
                evt = evt ? evt : window.event;
                $.ajax({
                    url:'@Url.Action("DeleteNotasExame", "CursosDisciplinas")',
                    method: 'post',
                    dataType: 'json',
                    data: { idExame:@Model.idexame},
                    success: function (dados) {
                        tabela.ajax.reload();
                        $(evt.target).prop("disabled", true);
                        $("#btinsertnotas").prop("disabled", false);
                        $("#msg").html("Foram eliminados " + dados.msg + " registos");
                    },
                    error: function (err) {
                        alert("Erro");
                    }

                });

            });

        });//DocumentLoad
```

```
</script>

 }
```

18 ANEXO H - LÓGICA DO CLIENTE <VIEW LISTAR AULAS>

```
@section aulasscripts{
    <script>


        $(function () {

            $("select").on("change", function (evt) {
                evt = evt ? evt : window.event;
                 var f= $(evt.target).parents("form");
                f.submit();
            });

            $("[name='lnkdel']").click(evt => {
                evt = evt ? evt : window.event;
                evt.preventDefault();
                alert(evt.target.id);
                $.ajax({
                    url:'@Url.Action("DeleteAula")',
                    type: 'get',
                    dataType: 'json',
                    data: { idaula: evt.target.id },
                    success: function (data) {
                        $("h3").html(data.msg);
                        $(evt.target).closest("tr").remove();

                    },
                    error: function () {
                        alert("Erro Ajax");
                    }
                });
            });
        });
    </script>

}
```

19 ANEXO I - LÓGICA DO CLIENTE <VIEW LISTAR DISCIPLINAS>

```
@section CursosDisciplinasScripts{
<script>
       $("[name='lnkdel']").click(function (evt) {
           evt = evt ? evt : window.event;
          // alert($(evt.target).attr("data-xpto"));
          // alert($(evt.target).data("xpto"));
          $.ajax({
               url: '@Url.Action("DeleteDisciplina","CursosDisciplinas")',
               type: 'get',
               dataType: 'json',
               data: {coddisc:$(evt.target).attr("id")},
               success: function (data) {
                   console.log(data.msg);
                   $("h3").html(data.msg);

                   if(data.msg=="ok")$(evt.target).closest("tr").remove();
               },
               error: function () { }

           });
       });
</script>
```

20 ANEXO J - LÓGICA DO CLIENTE < VIEW LISTAR EXAMES>

```
@section scripts{
    <script>
        $(function () {

            $("select").on("change", function (evt) {

                evt = evt ? evt : window.event;
                var f = $(evt.target).parents("form");
                f.submit();
            });

            $("[name='lnkdel']").click((evt) => {
                evt = evt ? evt : window.event;
                evt.preventDefault();
                if (!confirm("Quer apagar?")) return false;
                $.ajax({
                    url: '@Url.Action("DeleteExame")',
                    dataType: 'json',
                    data: { id: evt.target.id },
                    success: function (data) {
                        console.log(JSON.stringify(data));
                        if (data.msg == "Registo Eliminado com sucesso")
$(evt.target).closest('tr').remove();
                        $("h3").html(data.msg);

                    },
                    error: function () { alert("Ajax"); }
                });
            });//lnkdel



        });//onload
    </script>

}
```

21 ANEXO L - LÓGICA DO CLIENTE <VIEW USER ROLES>

```
@section Scripts{

    <script>
        $(function () {

           $("[name='lnkdel']").click(evt => {
                evt = evt ? evt : window.event;
                evt.preventDefault();
                let ok = confirm("Deseja eliminar User?");
                if (!ok) return false;
                $.ajax({
                    url:'@Url.Action("DeleteUser","Seguranca")',
                    type: 'get',
                    dataType: 'json',
                    data: { id: evt.target.id },
                    success: function (data) {
                        $("h3").html(data.msg);
                        $(evt.target).closest("tr").remove();

                    },
                    error: function () {
                        alert("Erro Ajax");
                    }
                });
            });


        });




    </script>

}
```

22 ANEXO M - LÓGICA DO CLIENTE <EDITAR ALUNOS TURMA>

```
@section EditarAlunosTurma{
 <script>
      function editpopup(num) {
         var url = "/Turmas/EditAlunoTurma?idturma=" + @Model.idturma + "&num=" + num;

         $("#zzzz").load(url, function () {
            $("#editalunosturma").modal("show");
         });

      }


   function popup() {
      var url = "/Turmas/AddAlunoTurma?idturma=" + @Model.idturma;
      $("#xxx").load(url,function () {
         $("#nifs").change(function (evt) {
            evt = evt ? evt : window.event;
            //alert(evt.target.id);
            $.ajax({
               url: '@Url.Action("DevolveNum", "Turmas")',
               dataType: 'json',
               method: 'post',
               data: { nif: $("#nifs").val() },
               success: function (data) {
                  console.log(data);
                  if (data.num == 0) {
                     $("#num").val(data.num);
```

```
                $('#num').attr('readonly', false);

            } else {
                $("#num").val(data.num);
                $('#num').attr('readonly', true);
            }

        },
        error: function () { alert("erro"); }
      });
    });
  $("#modalaluno2").modal("show");
  });

}//popup

var tabela2;
var id = @Model.idturma;

$(function () {
    $("#seek").val(@Model.idturma);
    $("#seek").change(function () {
      @*@ViewBag.seek=  $("#seek").val();*@
      location.href = '/Turmas/EditarAlunosTurma?id=' + $("#seek").val();
    });

    tabela2 = $("#alunitos").DataTable({
      "autoWidth": true,
      ajax: {
        url:'@Url.Action("GetAlunosTurma", "Turmas")',
        method: 'GET',
        datatype: 'json',
        data: { idt: id }
      },
      columns: [
        { data: "Num" },
```

```
                { data: "Nif" },
                { data: "Nome" },
                { data: "Idade"},
                {
                    data: "Num",
                    render: function (data,type, row) {
                        return "<button type='button'  data-num='" + row.Num + "'  class='btn btn-danger' onclick='LibertaAlunoTurma();'>Remover da Turma</button> <button type='button' class='btn btn-primary'    onclick='editpopup(" + row.Num+")'  >Editar AlunoTurma</button";
                    }
                },
            ]
        });
                                                    //del
        $("#btpopupalunodel").click(function (evt) {
            evt = evt ? evt : window.event;
            var nif = $("#nif").val();
            $.ajax({
                url: '@Url.Action("PopUpAlunoDel","Turmas")',
                dataType: 'json',
                method: 'post',
                data: {nif:nif},
                success: function (data) {
                    alert(data.msg);
                    if (data.msg == "Aluno Eliminado com sucesso") {
                        $("#modalaluno").modal("hide");
                    }
                    location.reload();
                },
                error: function () { alert("erro"); }
            });
        });

        //btpopupaluno (gravar aluno popup)
        $("#btgravarAluno").click(function (evt) {
            var aluno= {
```

```
                num: $("#num").val(),
                nif: $("#nif").val(),
                observacoes: $("#observacoes").val(),
                dataEntrada: $("#dataEntrada").val(),
                matriculaValidaAte: $("#matriculaValidaAte").val(),
                idturma: $("#idturma").val(),
                datainicio: $("#datainicio").val(),
                datafim: $("#datafim").val()
            }
             $.ajax({
                url: '@Url.Action("PopUpAluno")',
                dataType: 'json',
                method: 'post',
                data: {a:aluno},
                success: function (data) {
                   alert("Gravado com sucesso");
                   if (data.numero !=-1) {
                     //  $("#modalaluno").modal("hide");
                       var bt = "<input type='button' id='btinserir' value='Inserir Turma' class='btn
btn-primary' data-num='"+data.numero+"' onclick='btinsert();' />";
                       $("#bti").append(bt);
                   }
                },
                error: function () { alert("erro"); }
            });
        });




         }); //ajax

//_____LOAD_____
_____
```

```
function LibertaAlunoTurma(evt) {
    try {
        if (!confirm("Quer mesmo apagar?"))return;
        evt = evt ? evt : window.event;

        var vidturma =@Model.idturma;
        var vnum = $(evt.target).data("num");

        $.ajax({
            url: '@Url.Action("LibertaAlunoTurma", "Turmas")',
            type: 'post',
            dataType: 'json',

            data: { num:vnum ,idturma:vidturma},
            success: function (data) {
                if (data.msg == "ok") $(evt.target).closest("tr").remove();
                $("#msg").html("Registo Eliminado com sucesso");
            },
            error: function (erro) {
                alert(erro);
            }

        });
    } catch (e) {
        alert(e);
    }
}

//_____
_____

// function EditarAluno(nif) {
//     alert("zxzx");
```

```
    //      var url = "/Turmas/AddAlunoTurma?nif=" + nif;
    //      $("#modalalunobody").load(url, function () {
    //        $("#modalaluno").modal("show");
    //      });

    //}

//_____
_____

    function btdel(evt) {
        evt = evt ? evt : window.event;
        var idturma = $(evt.target).data("idturma");
        var num = $(evt.target).data("num");
        $.ajax({
            url:'@Url.Action("alunoturmadel", "Turmas")',
            method: 'post',
            dataType: 'json',
            data: { idturma: idturma, num:num },
            success: function (data) {
                console.log(data.msg);
                if (data.msg == 'ok') {
                    $(evt.target).closest("tr").remove();
                    $("#msg").text("Aluno Removido com sucesso");
                   // $("#modalaluno").modal("hide");
                    tabela2.ajax.reload();
                }
                else $("#msg").text(data.msg);
            },
            error: function (err) {
                alert("XXX");
            }
        });
    }
```

```
//_____
_____
        var oldlinha;
        function btedit(evt) {
            evt = evt ? evt : window.event;
            var linha = $(evt.target).closest("tr");
            oldlinha = linha;
            var tbody = $(evt.target).closest("tbody");
            turmaAluno = { num: linha.find("td:eq(0)").text(), idturma: linha.find("td:eq(1)").text(),
datainicio: linha.find("td:eq(3)").text(), datafim: linha.find("td:eq(4)").text() };
            var novalinha = "<tr id='edit'>";
            novalinha += "<td style = 'display: none;'><input type='text' value='"+turmaAluno.num
+"' id='txtnum'/></td>";
            novalinha += "<td style = 'display: none;'><input type='text'
value='"+turmaAluno.idturma+"' id='txtturma'/></td>";
            novalinha += "<td><select name='seledit' class='form-control
value="+turmaAluno.idturma +" '/></td>";
            novalinha += "<td><input type='text' id='txtdatainicio' class='form-control' value='"+
turmaAluno.datainicio +"' /></td>";
            novalinha += "<td><input type='text' id='txtdatafim' class='form-control' value='" +
turmaAluno.datafim +"' /></td>";
            novalinha += "<td><a class='btn btn-primary' onclick='fnbtsaveedit();'>ok</a>|";
            novalinha += "<a class='btn btn-warning' id='btcanceledit' onclick='fnbtcanceledit();'
>cncl</a></td>";
            novalinha += "</tr>";
            linha.replaceWith(novalinha);
            $.ajax({
              url:'@Url.Action("devolveTurmas", "Home")',
              method: 'post',
              dataType: 'json',
              success: function (data) {
                console.log(data);
                $.each(data, function (i, item) {
                  $("[name='seledit']").append($('<option>', {
                    value: item.idTurma,
                    text: item.codTurma
```

```
                }));

            });
            $("[name='seledit']").val(turmaAluno.idturma);
            $("#msg").text("Edição do Registo");
          },
          error: function (err) {
            alert("XXX");
          }
        });
        tbody.find("tr#edit").css({ 'background-color': 'lightgreen' });
        $("#msg").text("");
        $("#btinserir").prop('disabled', true);
      }


    function fnbtcanceledit(evt) {
      evt = evt ? evt : window.event;
      var linha = $(evt.target).closest("tr");
      linha.replaceWith(oldlinha);
      $("#btinserir").prop('disabled', false);
      $("#msg").text("");
    }

    function fnbtsaveedit(evt) {
        evt = evt ? evt : window.event;
        var linha = $(evt.target).closest("tr");
        var EditAluno = { num: linha.find("td:eq(0) #txtnum").val(), idturma: linha.find("td:eq(2) [name='seledit']").val(), datainicio: linha.find("td:eq(3) #txtdatainicio").val(), datafim: linha.find("td:eq(4) #txtdatafim").val() };
        console.log(turmaAluno);
         $.ajax({
          url:'@Url.Action("AlunoTurmaEdit", "Turmas")',
          method: 'post',
          dataType: 'json',
          data: { newAluno: EditAluno, oldAluno:turmaAluno},
```

```
                success: function (data) {
                    console.log(data);
                    var linhas = "";
                    $.each(data.turmasAluno, function (i, item) {
                        linhas += "<tr>";
                        linhas += "<td style = 'display: none;'>" + item.num + "</td>";
                        linhas += "<td style = 'display: none;'>" + item.idturma + "</td>";
                        linhas += "<td>"+item.codTurma +"</td>";
                        linhas += "<td>" + dataAtualFormatada(item.datainicio) + "</td>";
                        linhas += "<td>" + dataAtualFormatada(item.datafim) + "</td>";
                        linhas += "<td><a class='btn btn-danger' name='btdel' data-idturma='" + item.idturma + "' data-num='" + item.num + "' onclick='btdel();'>del</a> |";
                        linhas += "<a class='btn btn-success' name='btedit' data-idturma='" + item.idturma + "' data-num='" + item.num + "' onclick='btedit();'>edit</a></td>";
                        linhas += "</tr>";

                    });
                    $("#linhas").html("");
                    $("#linhas").append(linhas);
                    $("#btinserir").prop('disabled', false);
                    $("#msg").text(data.msg);
                },
                error: function (err) {
                    alert("XXX");
                }
            });
        }


        function dataAtualFormatada(data) {
            if (data == null) return "";
            var d = new Date(parseInt(data.replace("/Date(", "").replace(")/", ""), 10));
            return ("0" + d.getDate()).slice(-2) + "-" + ("0" + (d.getMonth() + 1)).slice(-2) + "-" + d.getFullYear();
        }
```

```
//_____
_____
        var novalinha = "<tr>";
        novalinha += "<td style = 'display: none;'><input type='text' id='txtnum'/></td>";
        novalinha += "<td style = 'display: none;'><input type='text' id='txtturma'/></td>";
        novalinha += "<td><select id='selinsert' class='form-control'/></td>";
        novalinha += "<td><input type='date' id='txtdatainicio' class='form-control' /></td>";
        novalinha += "<td><input type='date' id='txtdatafim' class='form-control' /></td>";
        novalinha += "<td><a class='btn btn-primary' onclick='btupdate();'>Update</a>|";
        novalinha += "<a class='btn btn-warning' id='btcancelnovo' onclick='btcancelnovo();'
>Cancel</a></td>";
        novalinha += "</tr>";

            function btinsert(evt) {
              evt = evt ? evt : window.event;
               $(evt.target).prop('disabled', true);
             $("#linhas").append(novalinha);
             $.ajax({
                url:'@Url.Action("devolveTurmas", "home")',
                method: 'post',
                dataType: 'json',
                success: function (data) {
                   console.log(data);
                   $.each(data, function (i, item) {
                      $('#selinsert').append($('<option>', {
                         value: item.idTurma,
                         text: item.codTurma
                      }));
                   });
                   $("#txtnum").val($(evt.target).data('num'));
                   $("#txtdatainicio").val();
                   $("#msg").text('Novo Registo');
                },
                error: function (err) {
                   alert("XXX");
```

```
            }
        });
    }


    function btupdate(evt) {
        evt = evt ? evt : window.event;
        var vidturma = $('#selinsert').val();
        var vnum = $('#txtnum').val();
        var vdatainicio = $('#txtdatainicio').val();
        var vdatafim = $('#txtdatafim').val();
        var valunoturma = { num: vnum, idturma: vidturma, datainicio: vdatainicio, datafim:
vdatafim };
        console.log(valunoturma);
        $.ajax({
            url:'@Url.Action("AlunoTurmaInsert", "Turmas")',
            method: 'post',
            dataType: 'json',
            data: {alunoturma:valunoturma},
            success: function (data) {
                console.log(data);
                var linhas = "";
                $.each(data.turmasAluno, function (i, item) {
                    linhas += "<tr>";
                    linhas += "<td style = 'display: none;'>" + item.num + "</td>";
                    linhas += "<td style = 'display: none;'>" + item.idturma + "</td>";
                    linhas += "<td>"+item.codTurma +"</td>";
                    linhas += "<td>" + dataAtualFormatada(item.datainicio) + "</td>";
                    linhas += "<td>" + dataAtualFormatada(item.datafim) + "</td>";
                    linhas += "<td><a class='btn btn-danger' name='btdel' data-idturma='" +
item.idturma + "' data-num='" + item.num + "' onclick='btdel();'>del</a> |";
                    linhas += "<a class='btn btn-success' name='btedit' data-idturma='" +
item.idturma + "' data-num='" + item.num + "' onclick='btedit();'>edit</a></td>";
                    linhas += "</tr>";

                });
```

```
                $("#linhas").html("");
                $("#linhas").append(linhas);
                $("#btinserir").prop('disabled', false);
                if (data.msg == 'ok') $("#msg").text('Registo inserido com sucesso');
                else $("#msg").text('Erro na inserção do registo');
            },
            error: function (err) {
                alert("XXX");
            }
        });

    }

    function btcancelnovo(evt) {
        try {
            evt = evt ? evt : window.event;
            $(evt.target).closest("tr").remove();
            $("#btinserir").prop('disabled', false);
            $("#msg").text('');
        } catch (e) {
            alert(e);
        }
    }

    </script>

}
```

23 ANEXO N- LÓGICA DO CLIENTES<LISTAR TURMA>

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}


@section tabelas{
    <script>

        var tabela;
        var PopUp;

        $(function () {
           $("#bt").click(function () {
                $("#tit").html("Editar Turma");
                var url = "/turmas/AddEdit";
                $("#modalturmabody").load(url, function () {
                    $("#modalturma").modal("show");
                });

        });

        tabela = $("#turminhas").DataTable({
                "autoWidth": true,
                ajax: {
                    url:'@Url.Action("GetTurmas","Turmas")',
                    method: 'GET',
                    datatype: 'json'

                },
                columns: [
                    { data: "idturma" },
                    { data: "nome"},
                    { data: "codCurso" },
                    { data: "anoEntrada" },
                    { data: "ano"  },
                    { data: "modalidade" },
                    {
                        data: "UC",
                        render: function (data) {
                            if (data == true) return "<input type='checkbox'
checked='checked' disabled/>";
                            return "<span> </span>";
```

```
                    }
                },
                { data: "codTurma" },
                {
                    data: "idturma",
                    render: function (data) {


                        return '<button type="button" name="del"
onclick="Apaga(' + data + ')" class="btn btn-danger" style="margin-right:5px"><span
class="glyphicon glyphicon-trash"></span></button><button type="button" name="del"
onclick="Edita(' + data +   ')" class="btn btn-success" ><span class="glyphicon
glyphicon-pencil"></span></button>';

                    },
                    orderable: false,

                }

            ]

           });

    });


    $(window).resize(function () {
        tabela.ajax.reload();
    });

    function Apaga (dado){
        $.ajax({
            url:'@Url.Action("DeleteTurma","Turmas")',
            type: 'get',
            datatype: 'json',
            data: { id: dado },
            success: function (data) {
                console.log(data);
                //if (data.msg == 'ok') tabela.ajax.reload();
                $("#msg").html(data.msg);
                tabela.ajax.reload();
            },
            error: function (erro) { alert(erro);}

        });
    }


    function Edita(dado) {
        $("#tit").html("Editar Turma");
        var url = "/turmas/AddEdit?id=" + dado;
        $("#modalturmabody").load(url, function () {
```

```
            $("#modalturma").modal("show");
        });
    }

    $("#btpopupturma").click(function (evt) {
         var turma = {
            idturma: $("#idturma").val(),
            codCurso: $("#codCurso").val(),
            anoEntrada: $("#anoEntrada").val(),
            modalidade: $("#modalidade").val(),
            nome:$("#nome").val(),
            UC: $("#UC").is(":checked")
        };
        $.ajax({
             url: '@Url.Action("AddEdit","Turmas")',
             dataType: 'json',
             method: 'post',
             data: { t: turma },
             success: function (data) {
                //if (data.msg == "Turma Gravada com sucesso") {
                //    $("#modalturma").modal("hide");
                //    tabela.ajax.reload();

                // }
                 $("#modalturma").modal("hide");
                 $("#msg").html(data.msg);
                 tabela.ajax.reload();
            },
            error: function (erro) { alert(erro); }

        });

    });

</script>

</script>

 }
```