UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

2025

Parte 2

RELATORIO

Laboratório de Planeamento e Desenvolvimento de Software

Grupo 6

Al80990 Bruno Alves

Al82339 Filipe Silva

Al79012 Liane Duarte

Al81311 Pedro Braz

Índice

Introdução

Este relatório corresponde à segunda etapa do Trabalho Prático da unidade curricular *Laboratório de Planeamento e Desenvolvimento de Software*. Após a definição do modelo de dados e da estrutura visual da aplicação na Parte 1, esta fase teve como principal objetivo a implementação funcional completa da aplicação, seguindo o padrão arquitetural **Model-View-ViewModel (MVVM)**.

A aplicação foi desenvolvida em C# com WPF, utilizando o Visual Studio 2022 e o .NET 9.0 como framework, integrando todas as funcionalidades previstas no protocolo fornecido. A gestão da informação é feita através de ficheiros XML, permitindo a persistência dos dados entre sessões. Ao longo desta etapa foram também implementadas validações de dados, mecanismos de interação entre as camadas MVVM e testes que asseguraram o correto funcionamento e robustez da plataforma.

Nas secções seguintes descreve-se a organização da aplicação, o modelo de dados e a forma como cada funcionalidade foi implementada de acordo com o padrão MVVM.

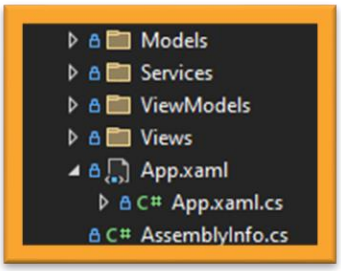
Objetivos etapa 2

A Etapa 2 teve como objetivo o desenvolvimento completo e funcional da aplicação de gestão de avaliações, respeitando os requisitos definidos no protocolo. Esta fase centrou-se na implementação da lógica da aplicação segundo o padrão arquitetural **Model-View-ViewModel (MVVM)**, com ênfase na separação clara entre dados, interface e lógica de apresentação.

Os principais objetivos foram:

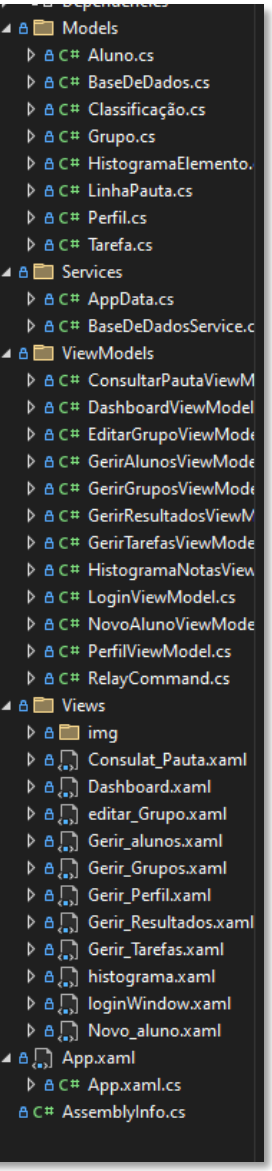
- **Implementar todas as funcionalidades previstas**, incluindo gestão de alunos, tarefas, grupos e classificações;

- **Persistir dados em ficheiros XML**, assegurando o armazenamento local das informações geridas;
- **Desenvolver a aplicação com base no padrão MVVM**, garantindo uma separação clara entre interface, lógica e dados;



Organização da Estrutura do Projeto

O projeto foi desenvolvido em **C# com WPF (.NET 9.0)** e encontra-se estruturado de forma modular, respeitando a separação de responsabilidades proposta pelo padrão **Model-View-ViewModel (MVVM)**. A estrutura de pastas está organizada da seguinte forma:



- **Models/**: Contém as classes que representam os dados da aplicação, como Aluno, Grupo, Tarefa, Classificação, Perfil, LinhaPauta, e HistogramaElemento.

- **Services/**: Reúne as classes responsáveis pela leitura, escrita e gestão dos dados persistentes, utilizando ficheiros **XML**, como BaseDeDadosService.cs e AppData.cs.

- **ViewModels/**: Implementa a camada intermediária entre os dados e a interface. Cada View tem um ViewModel correspondente, contendo as propriedades expostas à interface (ObservableCollection, string, etc.) e os comandos (ICommand) que suportam a lógica de interação. Esta pasta inclui, por exemplo, GerirAlunosViewModel.cs, LoginViewModel.cs e ConsultarPautaViewModel.cs, entre outros.

- **Views/**: Contém as interfaces gráficas da aplicação (.xaml), como Gerir_alunos.xaml, Dashboard.xaml, LoginWindow.xaml, Consultar_Pauta.xaml, entre outras. Cada uma delas está associada ao seu respetivo ViewModel.

Extensões ao Modelo de Dados da Etapa 1

Durante o desenvolvimento da Etapa 2, foi necessário **adicionar novos modelos de dados** com o objetivo de suportar melhor algumas funcionalidades.

As seguintes classes foram criadas e adicionadas à pasta Models/:

- **BaseDeDados.cs**

  Esta classe centraliza a informação da aplicação, agregando as listas de grupos, tarefas, classificações e o perfil do utilizador. Permite ler e guardar todos os dados num único ficheiro XML, facilitando a gestão da persistência.

- **HistogramaElemento.cs**

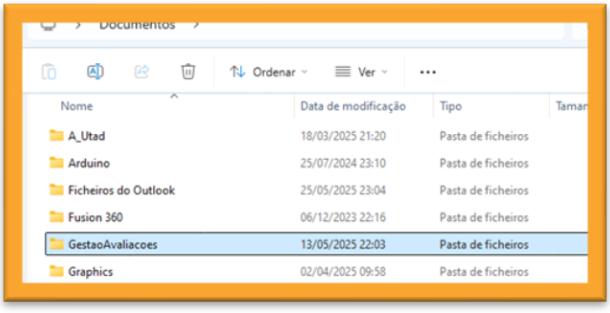
  Criada para dar suporte ao histograma de notas. Define as propriedades gráficas de cada barra (altura, cor, rótulo, frequência, etc.) e inclui também classes auxiliares como EixoLabel e LinhaGrid, que facilitam a construção visual do gráfico.

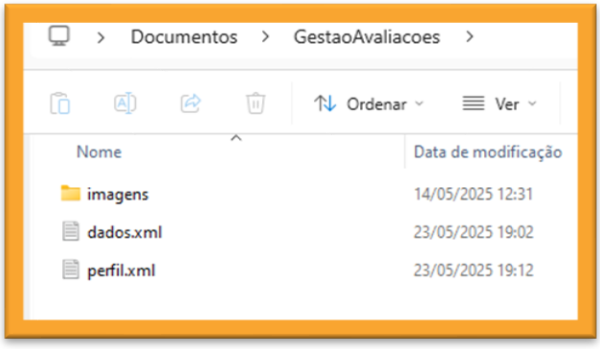
- **LinhaPauta.cs**

  Representa cada linha da pauta final de um aluno, contendo o número e nome do aluno, as notas por tarefa e a nota final ponderada. Esta estrutura permite gerar e visualizar a pauta de forma organizada e reutilizável.

Armazenamento de Dados

A aplicação guarda todos os dados de forma persistente no sistema de ficheiros local do utilizador, utilizando o formato **XML**. Para isso, é criada automaticamente uma pasta chamada **"GestaoAvaliacoes"** dentro da pasta **"Documentos"** do utilizador autenticado no Windows.





Ficheiros Utilizados:

- **dados.xml**
    - Contém a base de dados principal da aplicação, incluindo a lista de grupos, tarefas, classificações e alunos. Este ficheiro é gerado a partir da classe BaseDeDados.cs, que centraliza todas as coleções relevantes.

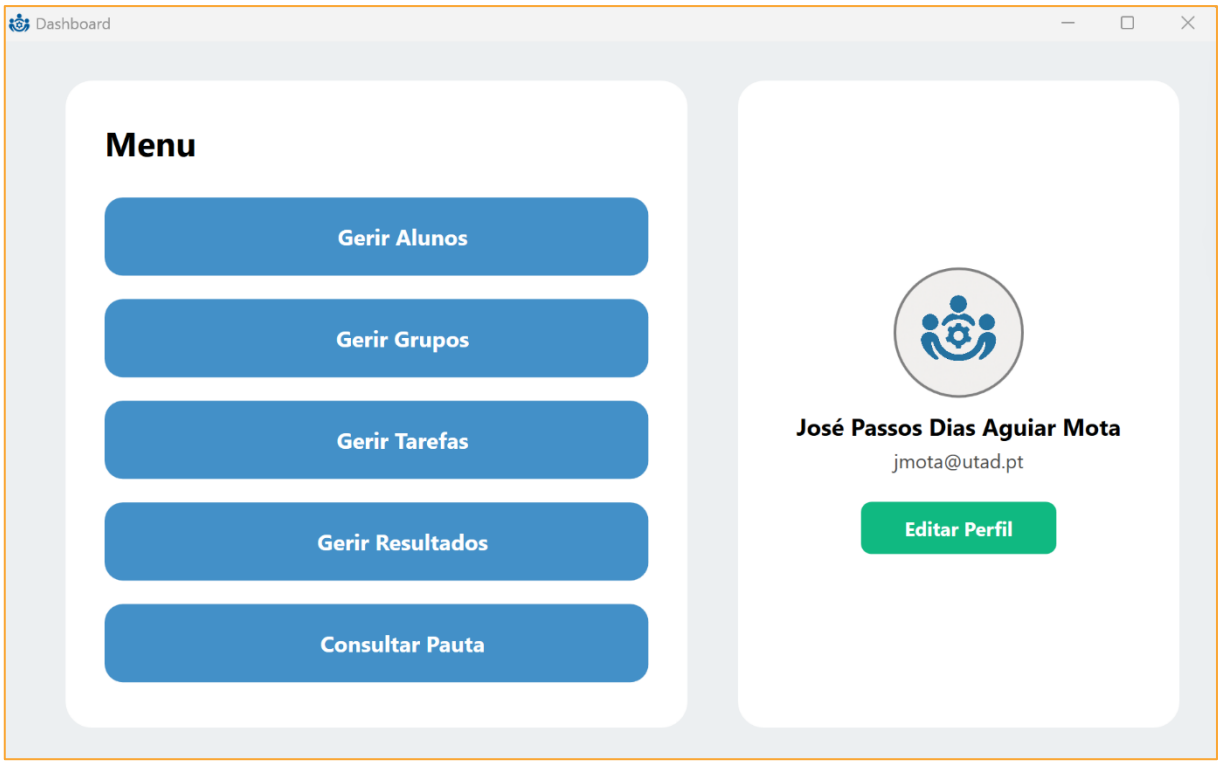
Exemplo ficheiro de dados em anexo no final do relatório (Anexo1)

Dividimos o xml em três elementos principais:

- **Grupos**: `<Grupos>` `</Grupos>`
    - Este elemento contém todos os grupos existentes na aplicação. Cada grupo é representado por um elemento `<Grupo>`, que inclui o seu **Id**, **Nome** e a lista de `<Alunos>`. Os alunos são identificados pelo seu Número, Nome e Email. Por defeito, os alunos importados ou criados sem grupo são automaticamente atribuídos ao **grupo com Id = 0**, que representa a categoria "Sem Grupo". Assim que são adicionados a um grupo específico, passam a ser incluídos no respetivo elemento `<Grupo>` no ficheiro XML.
- **Tarefas:** `<Tarefas>` `</Tarefas>`
    - Este elemento agrupa todas as tarefas registadas no sistema. Cada tarefa é representada por um elemento `<Tarefa>` no ficheiro XML.A sua função no armazenamento é permitir que as tarefas possam ser associadas a classificações posteriormente, mantendo os dados organizados e persistentes entre sessões. Os dados essenciais de cada tarefa — como o identificador, título, datas e peso — são serializados automaticamente pela aplicação aquando da gravação da base de dados.
- **Classificações:** `<Classificacoes>` `</Classificacoes>`
    - Este elemento armazena os registos de avaliação realizados pelos alunos nas diferentes tarefas. Cada elemento `<Classificacao>` representa a nota atribuída a um aluno numa tarefa específica, sendo identificada pelos respetivos **AlunoId** e **TarefaId**. Esta

separação permite que os dados das classificações fiquem desacoplados das entidades Aluno e Tarefa, tornando o sistema mais flexível e facilitando a leitura e atualização isolada das avaliações. As classificações são lidas e escritas no ficheiro XML sempre que o utilizador regista, edita ou remove uma nota.

Funcionalidades Implementadas

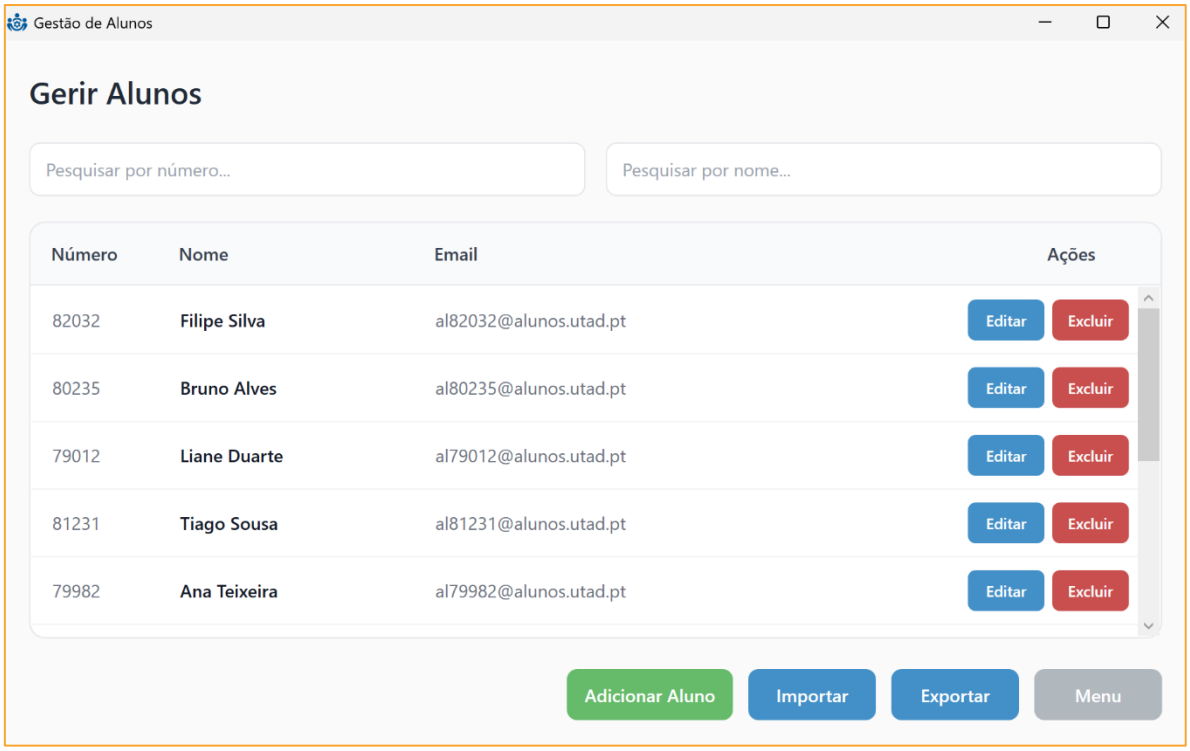


Gestão de Alunos

A funcionalidade de **Gestão de Alunos** permite adicionar, editar, remover e importar alunos para a base de dados da aplicação.

A janela principal associada a esta funcionalidade é a **Gerir_alunos.xaml**, à qual está ligada a classe **GerirAlunosViewModel.cs**, que implementa a lógica

associada à visualização e manipulação da lista de alunos. Cada ação é realizada através de **comandos** que interagem com o modelo Aluno e com o serviço de dados AppData.



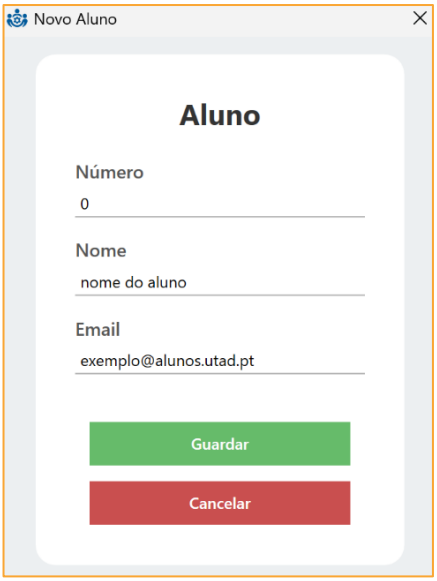
Adicionar novo aluno:

O botão “Adicionar Aluno” abre uma nova janela (Novo_aluno.xaml), onde o utilizador pode preencher os campos obrigatórios: número (obrigatório, numérico, único), nome (obrigatório) e email (obrigatório e necessário colocar um @).

A lógica associada encontra-se no ficheiro **Novo_aluno.xaml.cs**, que contém os manipuladores dos eventos dos botões. Quando o utilizador clica em “Guardar”, os dados introduzidos são validados:Se algum dos campos obrigatórios estiver vazio, é apresentada uma mensagem de erro.

- Se o número do aluno já existir (no caso de criação), a operação é bloqueada.

Caso os dados sejam válidos, o aluno é adicionado à lista em memória (AppData.Alunos) e os dados são imediatamente gravados no ficheiro XML (dados.xml) usando o método GuardarDados().

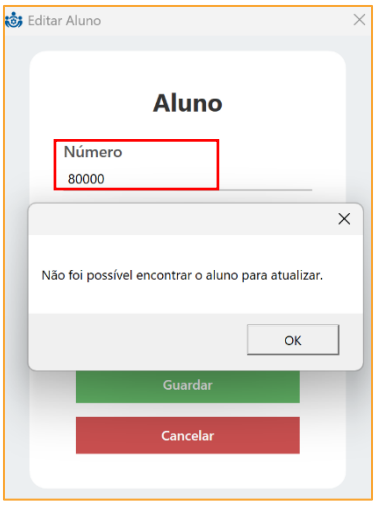


Editar aluno:

A edição de alunos é feita através da mesma janela utilizada na criação (Novo_aluno.xaml), com os campos já preenchidos com os dados do aluno selecionado. A lógica de edição encontra-se centralizada no NovoAlunoViewModel.cs, onde é verificado o modo de edição.

Durante a edição, o número do aluno é mostrado, mas não pode ser alterado. Esta restrição é essencial para garantir a consistência dos dados, pois as classificações estão associadas ao número do aluno. Caso o utilizador tente guardar alterações com um número modificado (por edição direta via código ou tentativa de manipulação), a operação é bloqueada e é apresentada uma mensagem de erro apropriada.

Assim, apenas o nome e o email do aluno podem ser atualizados. Após a confirmação, os dados são persistidos no ficheiro dados.xml usando a classe BaseDeDadosService.



Importar aluno:

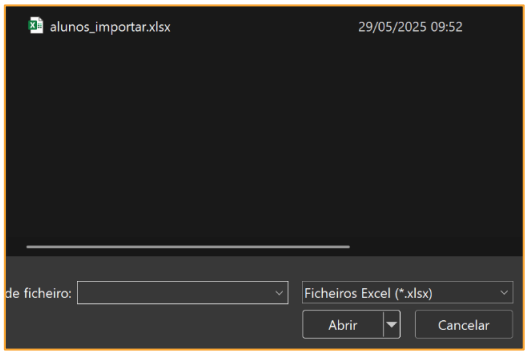
A aplicação permite ainda importar alunos a partir de ficheiros .xlsx, facilitando o carregamento em massa de dados. Esta funcionalidade está acessível através do botão “Importar” na interface da gestão de alunos e é gerida internamente pelo método Importar() na classe GerirAlunosViewModel.csrelatorio2GerirAlunosViewModel.

Durante o processo de importação, o sistema realiza as seguintes validações:

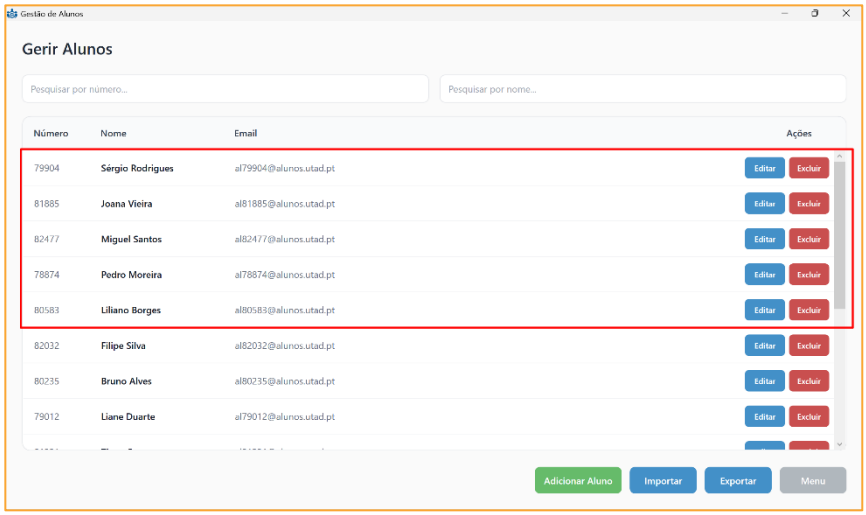
- O número deve ser numérico e único;
- O nome não pode estar vazio;
- O email deve conter o símbolo “@”;
- Alunos com número já existente na base de dados não são duplicados;
- Os alunos válidos são inseridos automaticamente no **Grupo 0 – Sem Grupo**, garantindo que nenhum aluno é deixado fora da estrutura de grupos da aplicaçãoBaseDeDadosService.

Após a leitura do ficheiro, a aplicação apresenta ao utilizador um resumo da operação, incluindo o número de alunos importados com sucesso e os erros encontrados (por exemplo, duplicados ou dados inválidos).

Este mecanismo foi implementado utilizando a biblioteca **EPPlus**, com licença não comercial, o que permite manipular ficheiros Excel sem recorrer ao Microsoft Office instaladoGerirAlunosViewModel.



Número	Nome	Email
79904	Sérgio Rodrigues	al79904@alunos.utad.pt
81885	Joana Vieira	al81885@alunos.utad.pt
82477	Miguel Santos	al82477@alunos.utad.pt
78874	Pedro Moreira	al78874@alunos.utad.pt
80583	Liliano Borges	al80583@alunos.utad.pt



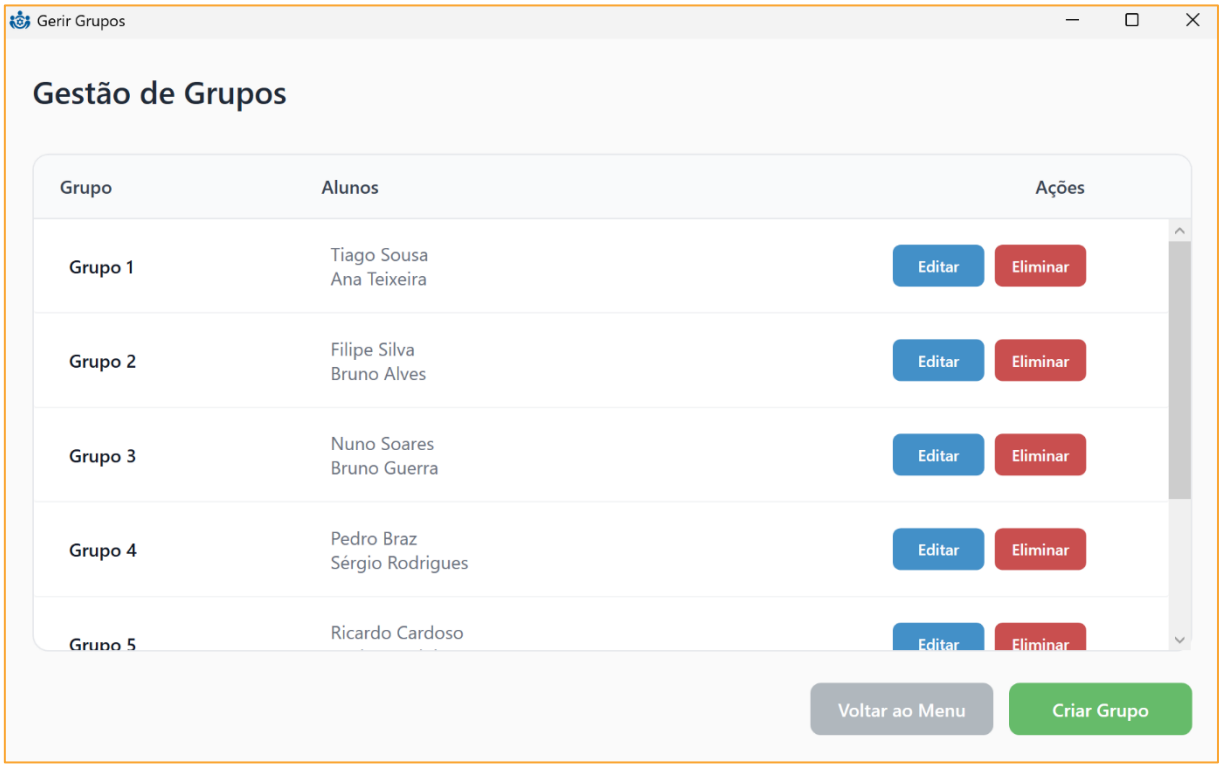
Exportar aluno:

A funcionalidade de exportação permite gerar um ficheiro Excel (.xlsx) com a lista de alunos atualmente visível na interface. Através do botão “Exportar”, o utilizador pode guardar localmente um documento que inclui as colunas **Número**, **Nome** e **Email**, facilitando a partilha e a consulta externa dos dados.

	A	B	C	D	E	F	G
1	Número	Nome	Email				
2	80235	Bruno Alves	al80235@alunos.utad.pt				
3	82032	Filipe Silva	al82032@alunos.utad.pt				
4	79904	Sérgio Rodrigues	al79904@alunos.utad.pt				
5	81885	Joana Vieira	al81885@alunos.utad.pt				
6	82477	Miguel Santos	al82477@alunos.utad.pt				
7	78874	Pedro Moreira	al78874@alunos.utad.pt				
8	80583	Liliano Borges	al80583@alunos.utad.pt				
9	79012	Liane Duarte	al79012@alunos.utad.pt				
10	81231	Tiago Sousa	al81231@alunos.utad.pt				
11	79982	Ana Teixeira	al79982@alunos.utad.pt				
12	82345	Ricardo Cardoso	al82345@alunos.utad.pt				
13	80389	Paulo Coutinho	al80389@alunos.utad.pt				
14	80634	Nuno Soares	al80634@alunos.utad.pt				
15	79009	Bruno Guerra	al79009@alunos.utad.pt				
16	81099	Pedro Braz	al81099@alunos.utad.pt				
17							

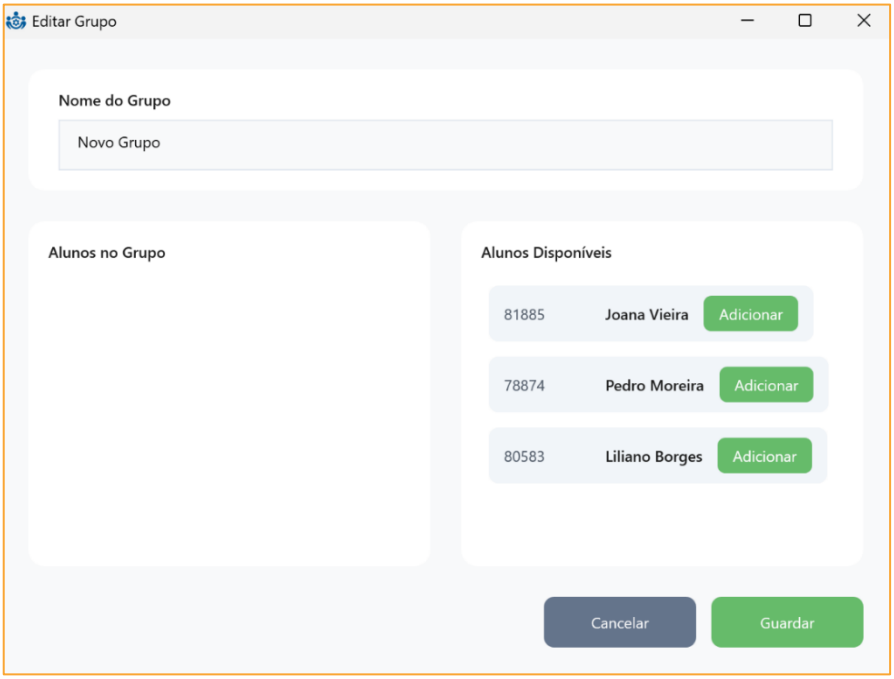
Gestão de Grupos

A funcionalidade de **Gestão de Grupos** permite criar, editar e eliminar grupos, bem como gerir os alunos associados a cada grupo. A aplicação assegura automaticamente que **cada aluno só pode pertencer a um único grupo**, conforme definido no protocolo.



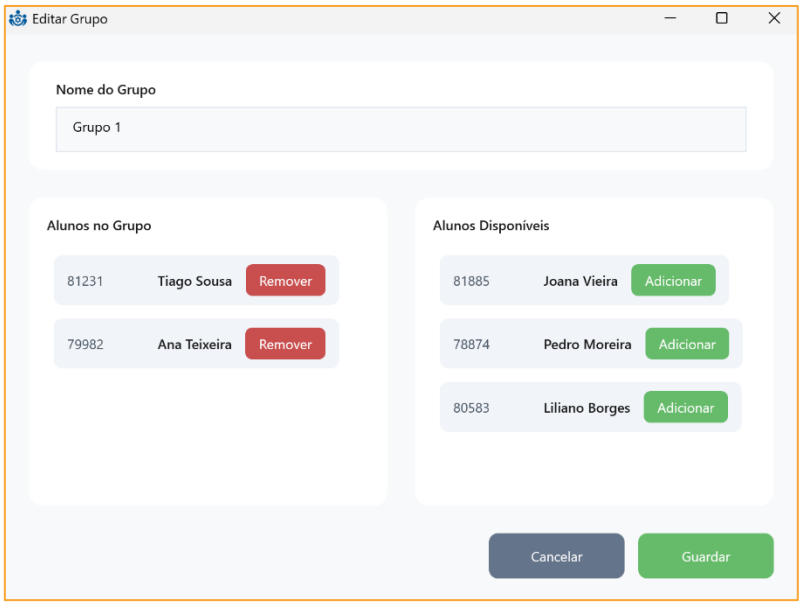
Criar Grupo:

Através do botão “Criar Grupo”, o utilizador abre uma nova janela onde pode definir o nome do grupo e selecionar os alunos a incluir. Os alunos disponíveis são automaticamente filtrados, excluindo aqueles que já pertencem a outros grupos.



Editar Grupo:

Permite alterar o nome do grupo e modificar a lista de alunos. A interface editar_Grupo.xaml apresenta dois painéis: um com os alunos já no grupo e outro com os alunos disponíveis. A lógica de verificação e atualização encontra-se no EditarGrupoViewModel.cs.



Eliminar Grupo:

Ao eliminar um grupo, os seus alunos são automaticamente movidos para o **Grupo 0 – Sem Grupo**. Este comportamento é tratado no método EliminarGrupo do GerirGruposViewModel.cs, garantindo que nenhum aluno fica fora de grupo.

```
<Grupo>
  <Id>0</Id>
  <Nome>Sem Grupo</Nome>
  <Alunos>
    <Aluno>
      <Numero>81885</Numero>
      <Nome>Joana Vieira</Nome>
      <Email>al81885@alunos.utad.pt</Email>
    </Aluno>
    <Aluno>
      <Numero>78874</Numero>
      <Nome>Pedro Moreira</Nome>
      <Email>al78874@alunos.utad.pt</Email>
    </Aluno>
    <Aluno>
      <Numero>80583</Numero>
      <Nome>Liliano Borges</Nome>
      <Email>al80583@alunos.utad.pt</Email>
    </Aluno>
  </Alunos>
```

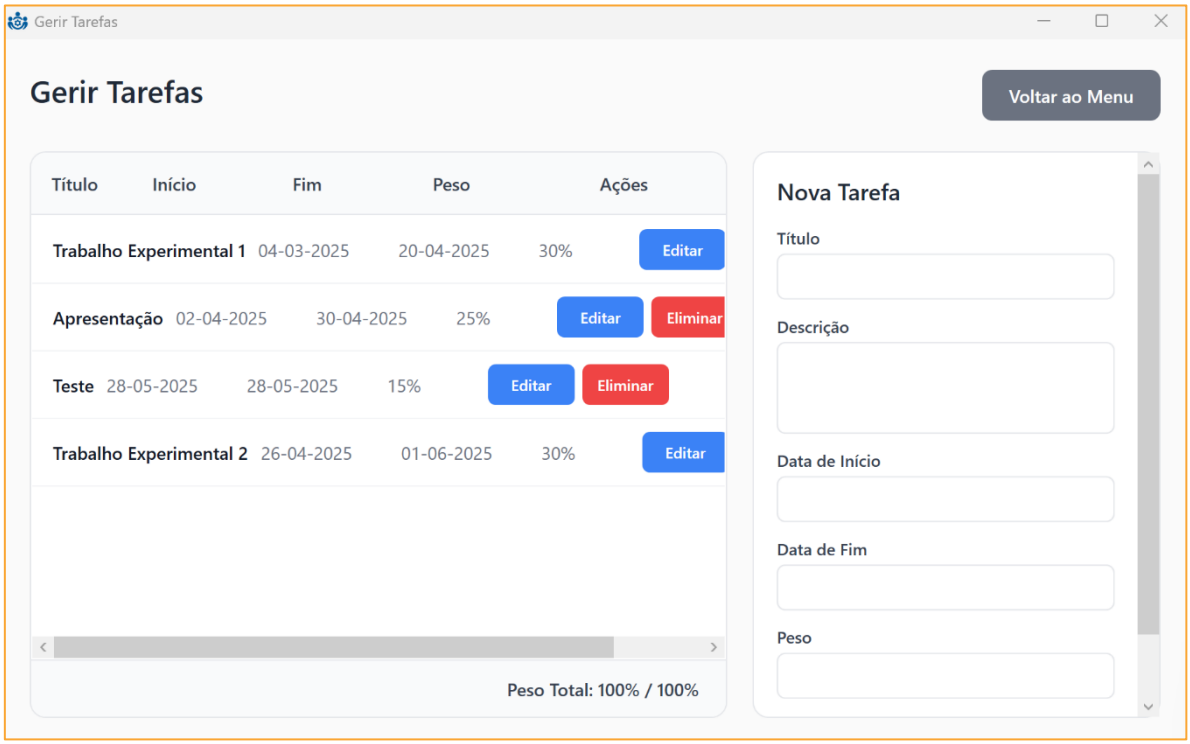
Validações:

- A aplicação impede a duplicação de alunos entre grupos.
- Durante o processo de guardar alterações, todos os grupos são limpos das referências aos alunos que vão ser movidos, garantindo que cada aluno pertence a um e apenas um grupo.
- O **Grupo 0** é sempre mantido pela aplicação (BaseDeDadosService.cs) e representa os alunos ainda não atribuídos.

Gestão de Tarefas

A funcionalidade de **Gestão de Tarefas** permite criar, editar e remover tarefas de avaliação, definindo os marcos que estruturam o progresso da unidade curricular. Estas tarefas são atribuídas **a todos os grupos** de forma global e afetam a média final dos alunos com base nos seus pesos relativos, conforme descrito no protocolo.

Cada tarefa inclui campos obrigatórios como **título**, **data de início**, **data de término** e **peso**, bem como um campo opcional de **descrição**.

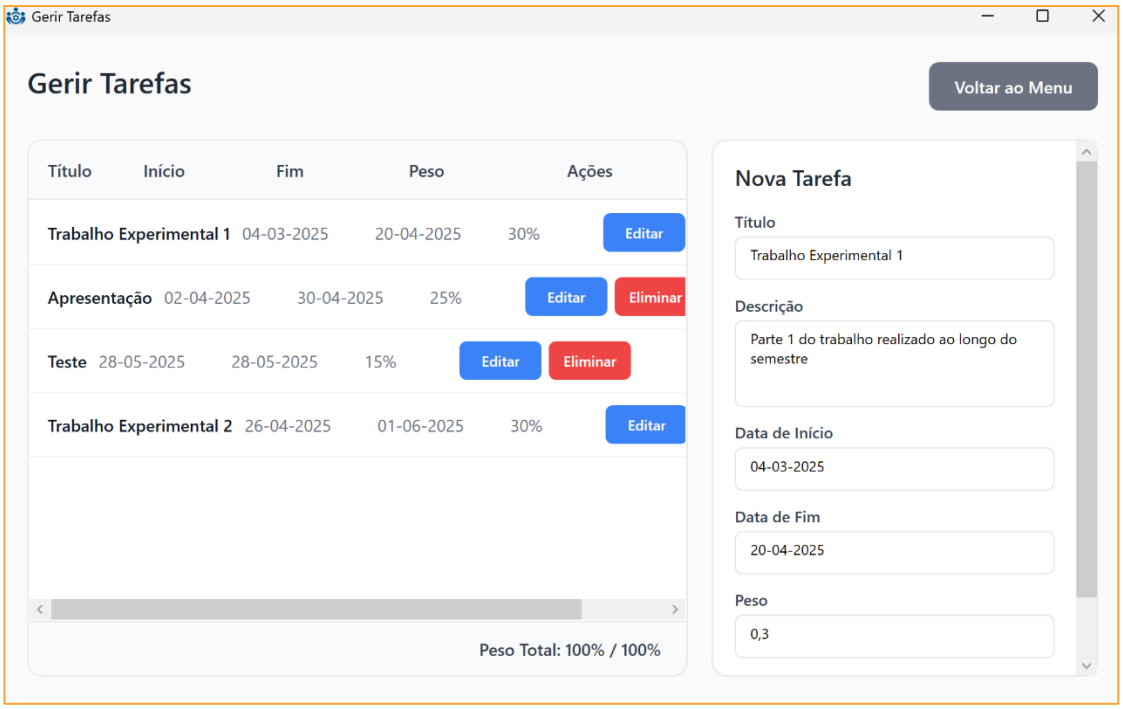


Criar tarefa:

A interface Gerir_Tarefas.xaml permite ao utilizador preencher um formulário com os campos da nova tarefa. A criação é feita com base nos dados inseridos, após validação de todos os campos obrigatórios. O ID da nova tarefa é gerado automaticamente, sendo sequencial com base nas tarefas já existentesGerirTarefasViewModel

Editar tarefa:

Ao selecionar uma tarefa e clicar em “Editar”, os campos do formulário são preenchidos com os dados da tarefa escolhida. A aplicação impede edições múltiplas simultâneas e solicita confirmação antes de iniciar a edição. Após guardar, a tarefa é atualizada no ficheiro XML.



Eliminar tarefa:

O botão “Eliminar” remove uma tarefa do sistema e, se existirem classificações associadas, estas são também eliminadas. O utilizador é sempre alertado com uma mensagem de confirmação, que indica o número de classificações a apagar.

Validações

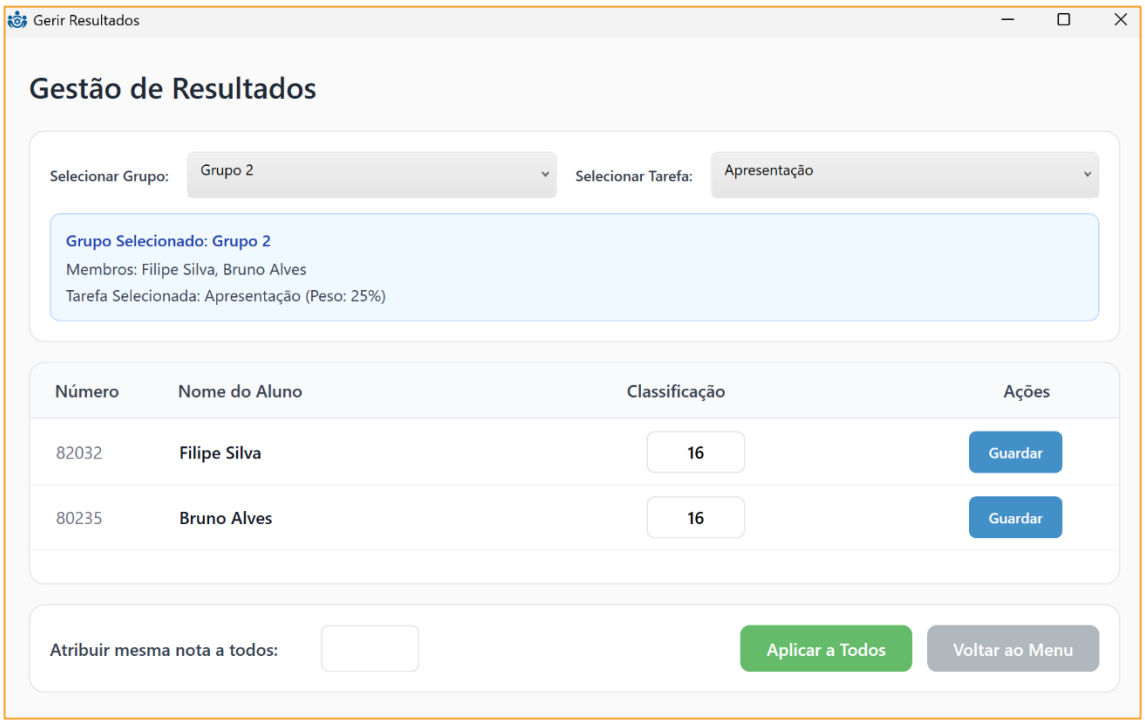
- As datas devem ser válidas (dd-mm-aaaa) e coerentes (início ≤ fim).
- O peso da tarefa deve estar entre 0 e 1, e é verificado se a soma dos pesos existentes mais o novo não ultrapassa 1 (100%)
- Em edição, a aplicação remove temporariamente a tarefa original para não contar duas vezes no somatório de pesos.

- Mensagens de erro e caixas de diálogo informativas orientam o utilizador na correção dos dados.

Gestão de Resultados

A funcionalidade de **Gestão de Resultados** permite ao utilizador atribuir e editar notas dos alunos relativamente a cada tarefa criada. Este registo é feito tendo por base os grupos formados e segue a premissa definida no protocolo: **a avaliação é normalmente atribuída ao grupo, mas armazenada individualmente por aluno e tarefa.**

As classificações são representadas pela classe Classificacao, que associa um AlunoId a um TarefaId, com o valor da nota obtida.



Seleção do grupo e tarefa:

O utilizador pode escolher um grupo e uma tarefa a partir de duas ComboBox.

Atribuição individual de notas:

Para cada aluno, é possível editar a nota diretamente na interface. O botão "Guardar" guarda a nota introduzida para o aluno selecionado, com validação de intervalo entre 0 e 20.

Atribuição em massa (nota para todos):

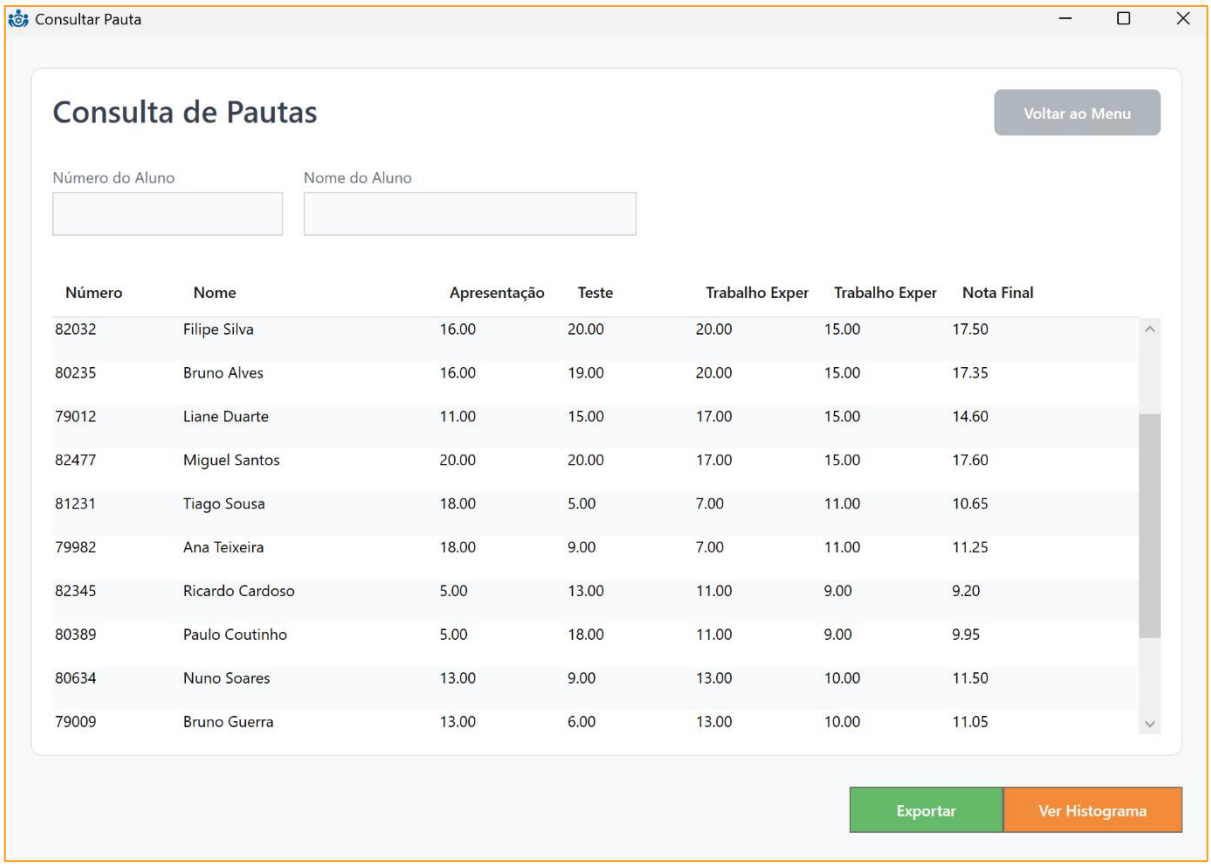
Um campo permite inserir uma nota comum a todos os alunos do grupo. Ao clicar em "Aplicar a todos", essa nota é automaticamente propagada a todos os alunos visíveis, com persistência imediata no ficheiro XML.

Validações

- O valor deve estar no intervalo [0, 20]; valores fora desse intervalo geram um MessageBox de aviso.
- A inserção de valores não numéricos no campo "Nota para Todos" também é impedida.

Consulta da Pauta

A funcionalidade de Consulta da Pauta oferece uma visão global e organizada do desempenho dos alunos em cada tarefa, bem como da média final ponderada. Está totalmente integrada com a funcionalidade de Histograma de Notas, permitindo uma análise visual da distribuição de resultados.



Estrutura da Pauta

Cada linha da pauta representa um aluno. A pauta inclui:

- Número e nome do aluno
- Notas atribuídas a cada tarefa
- Nota final ponderada

Os dados são representados pela classe LinhaPauta

Histograma das Notas

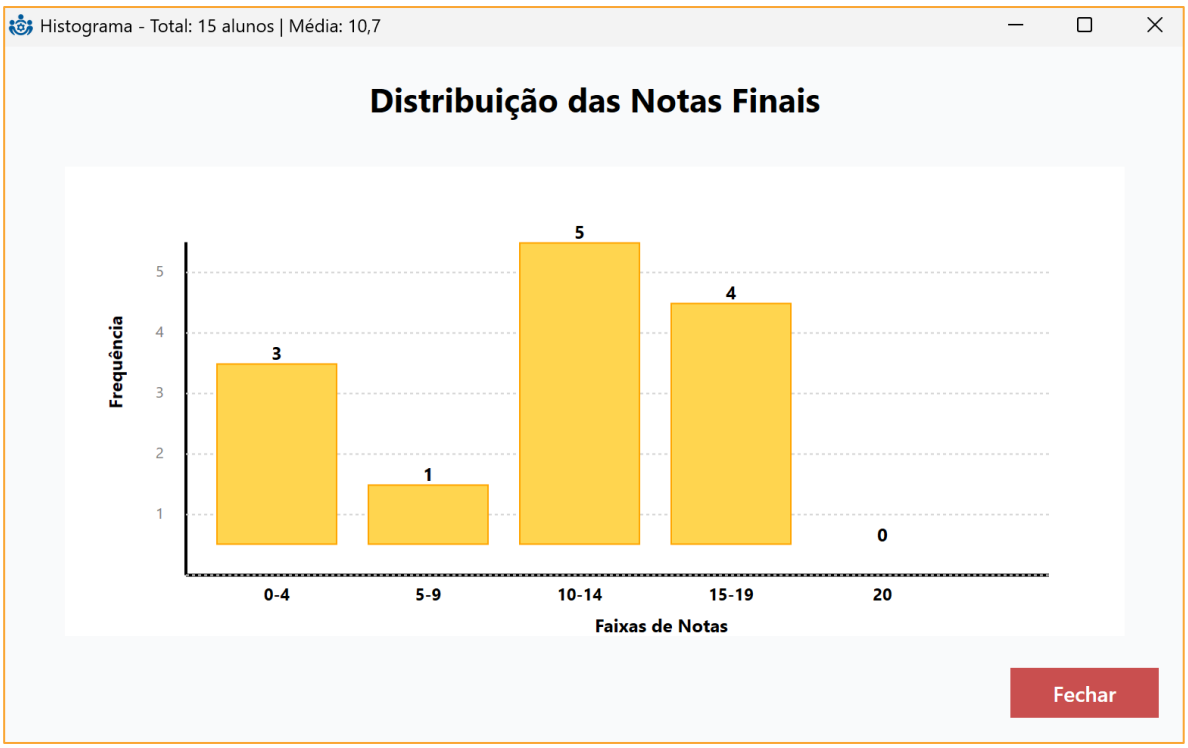
Ao clicar em "Ver Histograma", é aberta a janela HistogramaNotas.xaml, que apresenta um gráfico de barras construído com base na nota final de cada alunohistograma.xaml.

As notas são agrupadas nas seguintes faixas:

- 0–4
- 5–9
- 10–14
- 15–19
- 20

Cada faixa é representada por um objeto HistogramaElemento, que define:

- Posição X e Y
- Altura (frequência)
- Label da faixa
- Cor
- Texto de frequência visível acima da barra



Exportar

A interface da janela de consulta da pauta inclui um botão “Exportar”. Este botão está já ligado a um evento (Button_Click) que apresenta uma mensagem informativa a indicar que a funcionalidade será implementada no futuro.

Esta decisão foi tomada com base no facto de a exportação da pauta para ficheiro Excel estar definida como funcionalidade extra no protocolo do trabalho, destinada apenas a avaliação complementar ou em exame.

Perfil do Utilizador

A funcionalidade de **Gestão do Perfil do Utilizador** permite ao professor (utilizador da aplicação) configurar e atualizar os seus dados pessoais, garantindo um ambiente mais personalizado e alinhado com o contexto real da sala de aula.

A partir da janela Gerir_Perfil.xaml, o utilizador pode:

- **Editar o nome**
- **Editar o email**
- **Alterar a fotografia de perfil**
- **Guardar os dados atualizados**

Todos os campos estão ligados por binding ao PerfilViewModel, que por sua vez expõe uma instância da classe Perfil (modelo).

Alteração da Fotografia

O botão "Alterar Foto" permite ao utilizador selecionar uma nova imagem do disco (formatos suportados: .jpg, .jpeg, .png, .bmp). A imagem escolhida é automaticamente copiada para a pasta da aplicação (Documentos/GestaoAvaliacoes/imagens/foto_perfil/) e associada ao perfil.

O campo Fotografia é atualizado e a imagem é convertida para BitmapImage para exibição no Image do XAML.

Persistência dos dados

Os dados do perfil são guardados de forma independente num ficheiro perfil.xml, distinto do ficheiro dados.xml. A persistência é assegurada por AppData.GuardarPerfil(), utilizando serialização XML padrão.

Validações

- A aplicação valida a existência e o formato da imagem antes de aplicar a fotografia.
- Em caso de erro ao copiar a imagem ou guardar os dados, o utilizador é notificado com uma MessageBox clara e descritiva.

- A classe Perfil implementa INotifyPropertyChanged, permitindo que a interface seja atualizada automaticamente sempre que uma propriedade é modificadaPerfil.

Conclusão

A aplicação desenvolvida no âmbito da unidade curricular Laboratório de Planeamento e Desenvolvimento de Software permitiu consolidar os conhecimentos adquiridos ao longo do semestre, em particular no que diz respeito à aplicação do padrão arquitetural Model-View-ViewModel (MVVM), à separação de responsabilidades entre camadas e à utilização de tecnologias como WPF e serialização em XML.

Ao longo do projeto foi possível implementar de forma completa e funcional todas as funcionalidades previstas no protocolo, nomeadamente a gestão de alunos, grupos, tarefas, classificações, consulta da pauta e histograma de notas. A aplicação permite ainda ao utilizador gerir o seu próprio perfil, personalizando o nome, o email e a fotografia associada. Todo o sistema foi concebido com foco na clareza, na coerência dos dados e na usabilidade, garantindo que qualquer docente possa utilizar a plataforma de forma intuitiva e eficiente.

A estrutura modular da aplicação, suportada por modelos bem definidos, interfaces ligadas por data binding e comandos desacoplados, permitiu alcançar uma solução coesa e robusta. A implementação dos mecanismos de persistência assegura que todas as alterações são guardadas corretamente entre sessões, respeitando as boas práticas de organização de dados em ficheiros locais.

Em suma, o desenvolvimento desta aplicação representou um desafio prático enriquecedor, culminando numa solução funcional, clara e extensível, que cumpre integralmente os objetivos propostos e demonstra a aplicabilidade real dos conceitos explorados em aula.

Anexos

Anexo1.

```
<?xml version="1.0" encoding="utf-8"?>
<BaseDeDados>
 <Grupos>
  <Grupo>
   <Id>0</Id>
   <Nome>Sem Grupo</Nome>
   <Alunos>
    <Aluno>
     <Numero>10023</Numero>
     <Nome>Jorge</Nome>
     <Email>jorge@utad.pt</Email>
    </Aluno>
    <Aluno>
     <Numero>1234</Numero>
     <Nome>João da Silva</Nome>
     <Email>joao@email.com</Email>
    </Aluno>
   </Alunos>
  </Grupo>
 </Grupos>
 <Tarefas>
  <Tarefa>
   <Id>1</Id>
   <Titulo>Relatório Final</Titulo>
   <DataInicio>2025-05-01T00:00:00</DataInicio>
   <DataTermino>2025-05-15T00:00:00</DataTermino>
   <Peso>0.5</Peso>
  </Tarefa>
 </Tarefas>

 <Classificacoes>
  <Classificacao>
   <Alunoid>10023</Alunoid>
   <TarefaId>1</TarefaId>
   <Valor>18</Valor>
  </Classificacao>
  <Classificacao>
   <Alunoid>1234</Alunoid>
   <TarefaId>1</TarefaId>
   <Valor>15.5</Valor>
  </Classificacao>
 </Classificacoes>
</BaseDeDados>
```