p1g2 / Professor Joaquim Sousa Pinto

Base de dados EldenVault para jogo Elden Ring

Anderson Aoki Lourenço (108579), António Almeida (108250)

Índice

Introdução

Este projeto foi realizado no âmbito da disciplina de Base de Dados no ano de **2023/2024** e tem como objetivo recirar um sistema de gestão de dados do jogo **Elden Ring**, que contém toda a informação recorrente aos conteúdos do jogo.

Apresentaremos ao longo do relatório a maneira como implementámos a base de dados recorrendo ao **SQL Server Management Studio** e guardando as Queries executadas no servidor do **IEETA**.

Anexos:

- **Vídeo: https://youtu.be/tlUpXGmFLq0;**
- **Diagramas:** Pasta /Drawio (enviado dentro do .zip)
- **Login:**
    - username: admin
    - password: admin

Entidades

Item:

- Os itens abrangem tudo o que pode ser apanhado e utilizado no mundo de Elden Ring, desde armas e armaduras até feitiços e consumíveis.

Character:

- As personagens representam os seres que habitam o mundo de Elden Ring, sejam elas NPCs, jogadores ou outros seres importantes.

CraftingMaterials:

- Os materiais de crafting são os elementos base utilizados para a criação de itens.

Locations:

- As localizações representam as diversas áreas e regiões do mundo de Elden Ring.

Dungeon:

- As dungeons são masmorras desafiadoras que podem ser exploradas em Elden Ring.

Bosses:

- Os bosses são os inimigos mais poderosos de Elden Ring, exigindo estratégias e preparação para serem derrotados

Enemy:

- Os inimigos são os seres hostis que o jogador enfrenta em Elden Ring.

DER

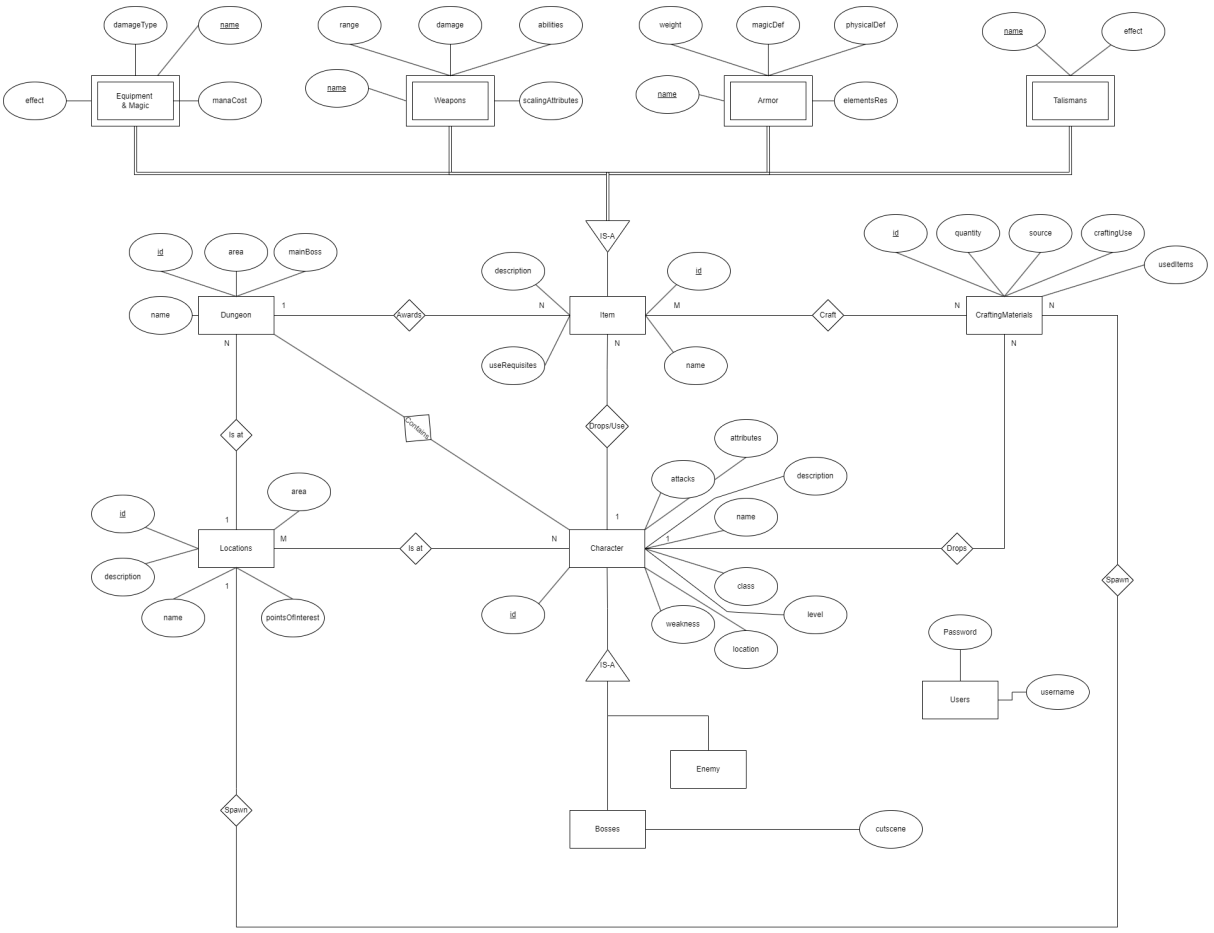


Fig1: Diagrama entidade/relacionamento

NOTA: Abrir ficheiro na pasta /Drawio (enviado dentro do .zip) para ver melhor o diagrama

Diagrama relacional

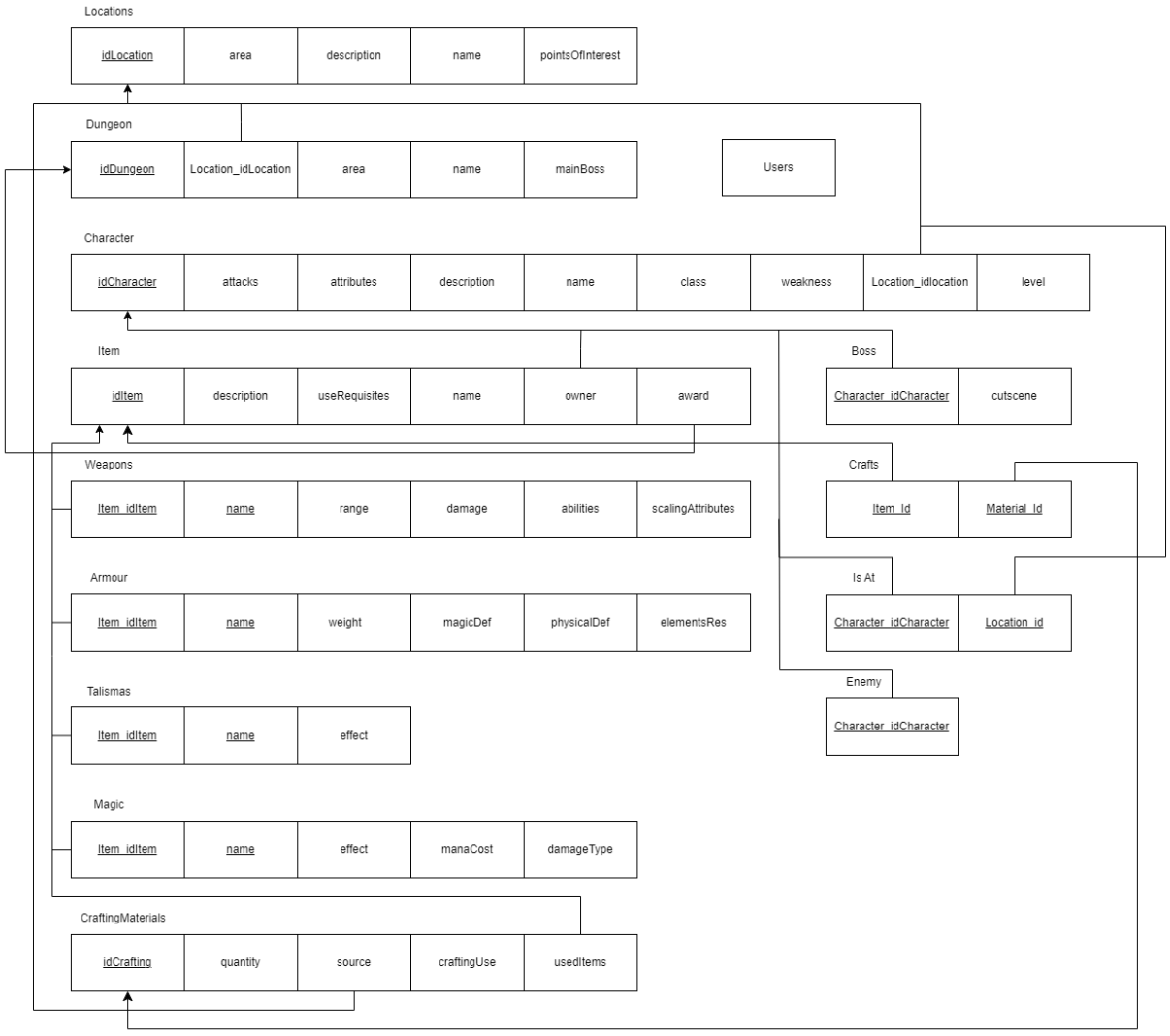


Fig2: Modelo Relacional

NOTA: Abrir ficheiro na pasta /Drawio (enviado dentro do .zip) para ver melhor o diagrama

Análise de requisitos

- Characters estão presentes em várias localidades;
- Um character larga vários items;
- Um character larga vários materiais de craft;
- Todos os “Bosses” são characters;
- Materiais aparecem em várias localidades;
- Uma localidade tem várias masmorras;
- Items podem ser encontrados em masmorras;
- Os items podem ser Magia/Armas/Armaduras/Talismãs;
- Masmorras contém characters;
- Materiais são usados para construir items.

DDL

O código **DDL** desempenhou um papel crucial na criação das tabelas na base de dados. Através dele, estabelecemos a definição e estrutura das tabelas selecionadas para o nosso projeto.

```
CREATE TABLE Users
(
   UserID INT IDENTITY(1,1) PRIMARY KEY,
   Username NVARCHAR(50) NOT NULL,
   Password VARBINARY(64) NOT NULL,
   Email NVARCHAR(100)
)
GO

CREATE TABLE Locations
(
   LocationID INT NOT NULL IDENTITY PRIMARY KEY,
   Area VARCHAR(512) NOT NULL,
   DESCRIPTION VARCHAR(1024) NOT NULL,
   Name VARCHAR(512) NOT NULL,
   PointsOfInterest VARCHAR(1024) NOT NULL,
)
GO

CREATE TABLE Dungeons
(
   -- Needs Locations
   DungeonID INT NOT NULL IDENTITY PRIMARY KEY,
   LocationID INT NOT NULL FOREIGN KEY REFERENCES Locations(LocationID),
   Area VARCHAR(512) NOT NULL,
   Name VARCHAR(512) NOT NULL,
   MainBoss VARCHAR(512) NOT NULL,
)
GO
```

```
CREATE TABLE Characters
(
   -- Needs Locations
   CharacterID INT NOT NULL IDENTITY PRIMARY KEY,
   Attacks VARCHAR(512) NOT NULL,
   Attributes VARCHAR(512) NOT NULL,
   DESCRIPTION VARCHAR(1024) NOT NULL,
   Name VARCHAR(512) NOT NULL,
   Class VARCHAR(512) NOT NULL,
   Weakness VARCHAR(512) NOT NULL,
   LocationID INT NOT NULL FOREIGN KEY REFERENCES Locations(LocationID),
   LEVEL INT NOT NULL,
)
GO
```

Fig 3: DDL Code

DML

A ferramenta **DML (Data Management Language)** é utilizada para inserir dados na nossa base de dados, exigindo a seleção adequada do tipo de dados para cada coluna. No entanto, a coerência dos dados com o jogo foi comprometida em prol de uma adição de dados mais simples.

```
SET IDENTITY_INSERT Locations ON;
INSERT INTO Locations (LocationID, Area, DESCRIPTION, Name, PointsOfInterest)
VALUES
(1,
'Fragment West Limgrave',
'Limgrave is a lush, expansive section of the Tenebrae Demesne.',
'LIMGRAVE',
'CHURCH OF DRAGON COMMUNION'),
(2,
'Fragment Weeping Peninsula',
'The peninsula, to Limgraves south, is named for its unceasing rainfall, redolent of lament.',
'WEEPING PENINSULA',
'CALLU BAPTISMAL CHURCH'),
(3,
'West Region of The Lands Between',
'With its shallow waters and vast wetlands, the region of Liurnia is beset with the gradual sinking of most of its landmass.',
'LIURNIA OF THE LAKES',
'BELLUM CHURCH'),
(4,
'South-East Region of The Lands Between',
'Caelid, known as the locale of the last battle between General Radahn and Malenia, Blade of Miquella.',
'CAELID',
'MINOR ERDTREE (CAELID)'),
...
SET IDENTITY_INSERT Locations OFF;
```

Fig 4:DML code

STORED PROCEDURES E TRANSACTIONS

As **Stored Procedures**, procedimentos pré-armazenados com um nome específico, não requerem recompilação a cada invocação. São armazenados em cache na memória na primeira execução, otimizando operações como adicionar, eliminar e editar dados, conforme exemplificado a seguir. É de realçar que é aqui que são usadas a maioria das nossas Transactions, procedimentos usado para alterar o mesmo campo de várias tabelas ao mesmo tempo, em segurança.

```
CREATE PROCEDURE AddEnemy
   (
   @CharacterID INT
)
AS
BEGIN
   DECLARE @error VARCHAR(512);
   BEGIN TRY
      INSERT INTO Enemies
      (CharacterID)
   VALUES
      (@CharacterID);
   END TRY
   BEGIN CATCH
      SELECT @error = ERROR_MESSAGE();
      SET @error = 'Error, could not add enemy to database. Value added incorrectly.'
      RAISERROR(@error, 16, 1);
   END CATCH
END
```

Fig 5: Exemplo de Add Store Procedure

```
DROP PROCEDURE DeleteCraftingMaterial;

GO

CREATE PROCEDURE DeleteCraftingMaterial(@ID_CraftingMaterial INT)

AS

BEGIN
   DECLARE @verification INT;
   DECLARE @error VARCHAR(100);

   SET @verification = (SELECT dbo.check_CraftingMaterialID(@ID_CraftingMaterial));

   IF (@verification = 0)
      BEGIN
      SET @error = 'CraftingMaterialID does not exist, please check the ID and try again!';
      RAISERROR (@error, 16, 1);
   END
      ELSE
      BEGIN
      BEGIN TRY
            BEGIN TRAN
               DELETE FROM Crafts WHERE CraftingMaterialID = @ID_CraftingMaterial;
               DELETE FROM CraftingMaterials WHERE CraftingMaterialID = @ID_CraftingMaterial;
            COMMIT TRAN
         END TRY
         BEGIN CATCH
            ROLLBACK TRAN
            SELECT @error = ERROR_MESSAGE();
            SET @error = 'Error, could not delete crafting material from database. Value deleted
incorrectly.';
            RAISERROR (@error, 16, 1);
         END CATCH
   END
END
```

Fig 6: Exemplo de Delete Store Procedures com Transactions

```
CREATE PROCEDURE EditDungeon
   (
   @ID_Dungeon INT,
   @LocationID INT,
   @Area VARCHAR(512),
   @Name VARCHAR(512),
   @MainBoss VARCHAR(512)
)

AS

BEGIN
   DECLARE @verification INT;
   DECLARE @error VARCHAR(100);

   SET @verification = (SELECT dbo.check_DungeonID(@ID_Dungeon));

   IF (@verification = 0)
      BEGIN
      SET @error = 'DungeonID does not exist, please check the ID and try again';
      RAISERROR (@error, 16, 1);
   END
   ELSE
      SET NOCOUNT ON;
   BEGIN
      BEGIN TRY
            BEGIN TRAN

               UPDATE Dungeons
               SET
                  LocationID = @LocationID,
                  Area = @Area,
                  Name = @Name,
                  MainBoss = @MainBoss
               WHERE
                  DungeonID = @ID_Dungeon;
            COMMIT TRAN
         END TRY
         BEGIN CATCH
            Rollback TRAN
```

```
            SELECT @error = ERROR_MESSAGE();
            SET @error =  'Error, could not edit dungeon in database. Value edited incorrectly.'
            RAISERROR (@error, 16,1);
        END CATCH
    END
END
```

Fig 7: Exemplo de Edit Store Procedures com Transactions

VIEW

As **views** podem ser utilizadas como fonte de dados para instruções SQL. Além de as visualizarmos na interface gráfica, desenvolvemos-nas para facilitar consultas personalizadas, permitindo a visualização de dados específicos, tal como ilustrado nos excertos de código seguintes:

```
CREATE VIEW CharactersView_Table
AS
  SELECT
    Characters.CharacterID AS ID,
    Characters.Name,
    Characters.DESCRIPTION as Description,
    Characters.Class,
    Characters.Attacks,
    Characters.Attributes,
    Characters.Weakness,
    Characters.LEVEL as Level,
    Locations.Area,
    Locations.Name AS LocationName
  FROM Characters
    JOIN Locations ON Characters.LocationID = Locations.LocationID
GO
```

Fig 8: Exemplo de Views

UDF

As **UDFs**, ou funções definidas pelo usuário, podem receber argumentos de entrada e retornar valores. Aproveitamos esta capacidade para realizar verificações, como a existência de atributos (principalmente chaves primárias) numa tabela, e para obter valores específicos, como ilustrado nos excertos de código seguintes.

```
CREATE FUNCTION check_DungeonID (@ID_Dungeon INT) RETURNS INT
AS
  BEGIN
    DECLARE @counter INT
    SELECT @counter=COUNT(1) FROM Dungeons WHERE DungeonID=@ID_Dungeon
    RETURN @counter
  END
```

Fig 9: Exemplo de UDF

TRIGGERS

Os **triggers** são acionados em situações específicas de manipulação de dados. Eles são "disparados" quando uma ação pré-definida é realizada. Utilizamos apenas triggers do tipo AFTER INSERT para impedir que sejam criados Bosses com level inferior a 100.

```
CREATE TRIGGER trg_RemoveLowLevelCharacter
ON Bosses
AFTER INSERT
AS
BEGIN
   -- Delete rows where the character's level is less than 100
   DELETE b
   FROM Bosses b
   JOIN Characters c ON b.CharacterID = c.CharacterID
   WHERE c.Level < 100;
   RETURN;
END;
```

Fig 10: Exemplo de Triggers

INDEXES

Para aprimorar a busca por informações, utilizamos **índices** na nossa base de dados. Essa estratégia, mesmo em bases de dados pequenas, demonstrou ser eficaz para localizar tabelas por ID e nome.

```
CREATE INDEX IX_Locations_Id ON Locations(LocationID, Name);
GO
CREATE INDEX IX_Dungeons_Id ON Dungeons(DungeonID, Name);
GO
CREATE INDEX IX_Characters_Id ON Characters(CharacterID, Name);
GO
CREATE INDEX IX_CraftingMaterials_Id ON CraftingMaterials(CraftingMaterialID, Name);
```

Fig 11: Exemplo de Indexes

Paging

Para o tamanho da nossa base de dados, a existência de **Paging** não se vê muito útil, no entanto, para base de dados maiores, a receber apenas X linhas de uma só vez é benéfico para a otimização. Implementámos paging numa das tabelas, para demonstrar a sua utilização.
NOTA: (adicionado depois da apresentação).

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Items_Table ORDER BY ID OFFSET
@Offset ROWS FETCH NEXT @PageSize ROWS ONLY", CN);
cmd.Parameters.AddWithValue("@Offset", offset);
cmd.Parameters.AddWithValue("@PageSize", pageSize);

DataTable detailsTable = new DataTable();
SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(cmd);
cmd.ExecuteNonQuery();
detailsTable.Clear();
sqlDataAdapter.Fill(detailsTable);

ShowTableInfo.DataSource = detailsTable;
ShowTableInfo.AutoResizeRows();
ShowTableInfo.AutoResizeColumns();
ShowTableInfo.Visible = true;
```

Fig 12: Exemplo de Paging

CURSOR

Os **cursores** permitem a leitura sequencial de dados, o que os torna ideais para visualizar o número de Enemies com o mesmo ataque. Implementamos essa funcionalidade, atualizando o valor e mostrando-o em uma caixa de mensagem através de um botão, sempre que é escrito o ataque.

```
CREATE PROCEDURE GetEnemiesWithAttack(@Attack VARCHAR(512), @AttackTotal INT
OUTPUT)
AS
BEGIN
   DECLARE @EnemyCount INT

   DECLARE enemyCursor CURSOR FOR
   SELECT COUNT(*) AS EnemyCount
   FROM Enemies e
   JOIN Characters c ON e.CharacterID = c.CharacterID
   WHERE c.Attacks = @Attack

   OPEN enemyCursor
   FETCH NEXT FROM enemyCursor INTO @EnemyCount

   IF @@FETCH_STATUS = 0
   BEGIN
      SET @AttackTotal = @EnemyCount
   END

   CLOSE enemyCursor
   DEALLOCATE enemyCursor
END
```

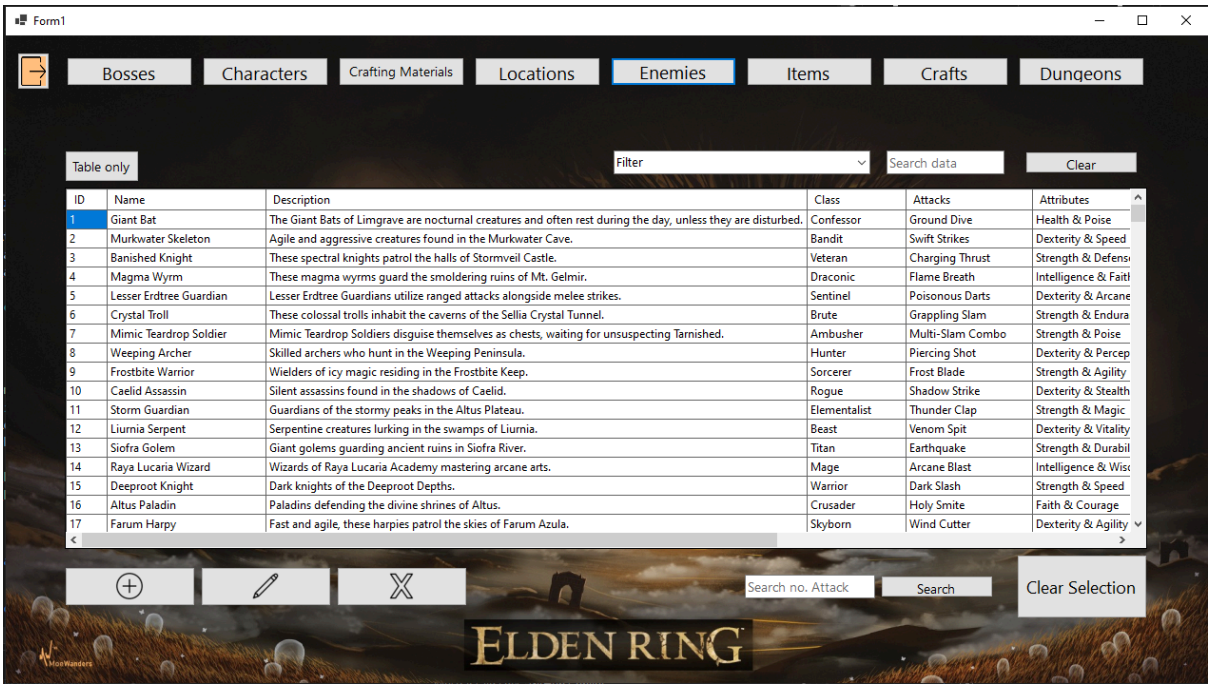
Fig 13: Exemplo de Cursor

INTERFACE GRÁFICA

A interface foi desenvolvida no ambiente **Visual Studio**, utilizando a ferramenta **Windows Forms** e linguagem de programação **C#**.
O efeito de diferents páginas foi conseguido através de panels sovrepostos que são exibidos ao clicar no botão do tem específico.

Quanto às tabelas, os dados são exibidos através de **DataGridViews** permitindo a organização dos valores da tabela pela respetiva coluna.
Para além disso, todos os paneis têm no canto superior direito um conjunto de uma **ComboBox** com os nomes das colunas da tabela, uma **TextBox** para os dados de pesquisa e um botão de **limpar filtros** que auxiliam à pesquisa de dados por categoria.

Para permitir a manipulação de dados, todos os paneis têm um conjunto de botões de adicionar, eliminar e editar a respetiva tabela(caso estando em login como **admin**). Ambos adicionar e editar abrem uma nova janela para efetuar as operações desejadas.
Por fim, ao realizar as operações de adicionar, eliminar ou editar. caso ocorra algum erro este será exibido numa **MessageBox**. bem como caso a operação seja sucedida é exibida da mesma forma uma mensagem informando que a ação foi realizada com sucesso sendo a tabela exibida ao utilizador atualizada com os respetivos dados.
Vídeo completo em https://youtu.be/tlUpXGmFLq0.



ID	Name	Description	Class	Attacks	Attributes
1	Giant Bat	The Giant Bats of Limgrave are nocturnal creatures and often rest during the day, unless they are disturbed.	Confessor	Ground Dive	Health & Poise
2	Murkwater Skeleton	Agile and aggressive creatures found in the Murkwater Cave.	Bandit	Swift Strikes	Dexterity & Speed
3	Banished Knight	These spectral knights patrol the halls of Stormveil Castle.	Veteran	Charging Thrust	Strength & Defens
4	Magma Wyrm	These magma wyrms guard the smoldering ruins of Mt. Gelmir.	Draconic	Flame Breath	Intelligence & Faith
5	Lesser Erdtree Guardian	Lesser Erdtree Guardians utilize ranged attacks alongside melee strikes.	Sentinel	Poisonous Darts	Dexterity & Arcane
6	Crystal Troll	These colossal trolls inhabit the caverns of the Sellia Crystal Tunnel.	Brute	Grappling Slam	Strength & Endura
7	Mimic Teardrop Soldier	Mimic Teardrop Soldiers disguise themselves as chests, waiting for unsuspecting Tarnished.	Ambusher	Multi-Slam Combo	Strength & Poise
8	Weeping Archer	Skilled archers who hunt in the Weeping Peninsula.	Hunter	Piercing Shot	Dexterity & Percep
9	Frostbite Warrior	Wielders of icy magic residing in the Frostbite Keep.	Sorcerer	Frost Blade	Strength & Agility
10	Caelid Assassin	Silent assassins found in the shadows of Caelid.	Rogue	Shadow Strike	Dexterity & Stealth
11	Storm Guardian	Guardians of the stormy peaks in the Altus Plateau.	Elementalist	Thunder Clap	Strength & Magic
12	Liurnia Serpent	Serpentine creatures lurking in the swamps of Liurnia.	Beast	Venom Spit	Dexterity & Vitality
13	Siofra Golem	Giant golems guarding ancient ruins in Siofra River.	Titan	Earthquake	Strength & Durabil
14	Raya Lucaria Wizard	Wizards of Raya Lucaria Academy mastering arcane arts.	Mage	Arcane Blast	Intelligence & Wisc
15	Deeproot Knight	Dark knights of the Deeproot Depths.	Warrior	Dark Slash	Strength & Speed
16	Altus Paladin	Paladins defending the divine shrines of Altus.	Crusader	Holy Smite	Faith & Courage
17	Farum Harpy	Fast and agile, these harpies patrol the skies of Farum Azula.	Skyborn	Wind Cutter	Dexterity & Agility

Fig 14: Menu inicial

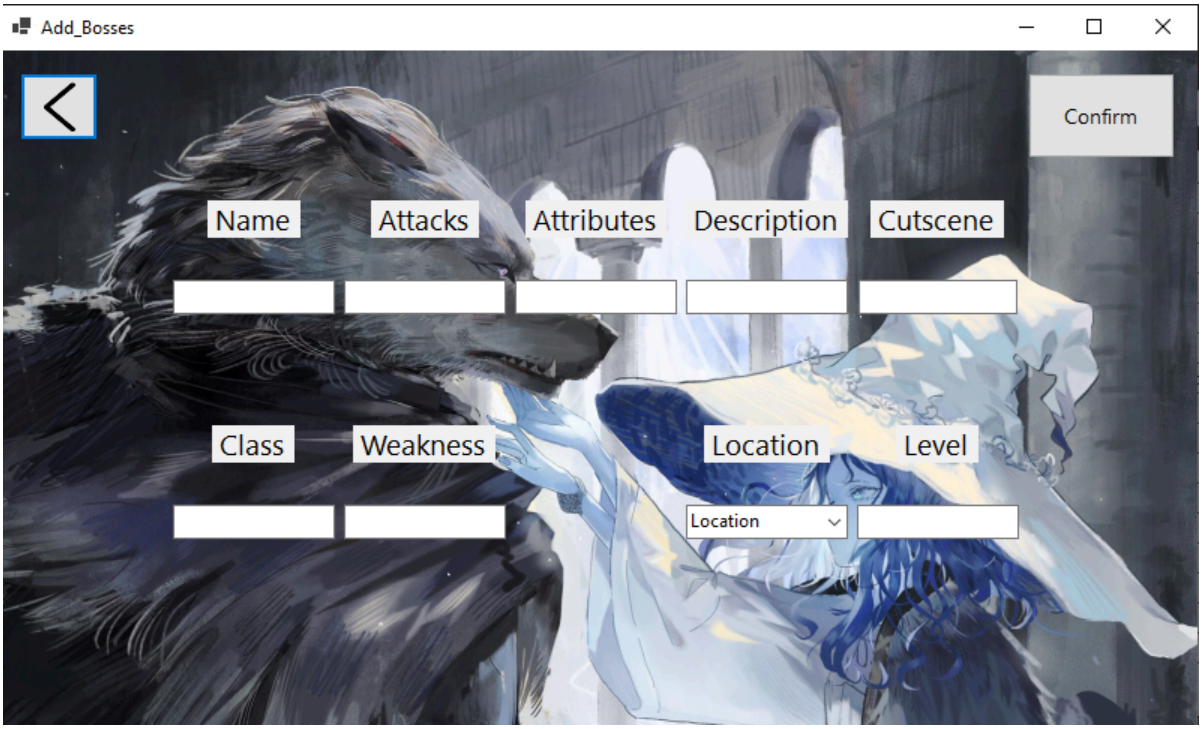


Fig 15: Exemplo da janela add para a tabela Characters

Conclusão

Em conclusão. este projeto apresentou uma solução que nós achamos minimamente interessante para a gestão de dados Elden Ring. utilizando a **SQL** e a **C#** como ferramentas de desenvolvimento.

A **SQL** mostrou ser crucial na construção de uma base de dados eficiente e segura. facilitando o armazenamento, organização e recuperação de informações, assegurando, assim, a integridade e a precisão dos dados. Por outro lado, a linguagem **C#** permitiu a construção de uma interface de utilizador amigável e intuitiva. além de integrar de maneira eficaz as funcionalidades da base de dados, facilitando a interação do utilizador final com o sistema.

Consideramos ter cumprido os objetivos a que nos propusemos para este projeto e agradecemos aos professores envolvidos por nos guiarem na realização do mesmo, tanto ao professor Carlos Costa na componente teórica como ao professor Joaquim Sousa Pinto na componente prática.