universidade de aveiro

2022/2023

MINEBASE

PROJETO DE BASE DE DADOS

LILIANA RIBEIRO, 108713

LIA CARDOSO, 107548

P8

ÍNDICE

INTRODUÇÃO

Minecraft é um jogo de mundo aberto que se foca em duas ações principais: minerar blocos e construir novas estruturas com os mesmos. O jogo é extremamente popular, contando com mais de 120 milhões de jogadores, e está constantemente a sofrer *updates* que trazem novos elementos para os utilizadores explorarem.

O nosso projeto final de "Base de Dados" consiste em criar uma base de dados que contém informações acerca dos elementos do jogo como por exemplo blocos, mobs (monstros e animais), armas, comida e itens. Além disso, para tornar a premissa do projeto mais interessante decidimos implementar algumas dinâmicas do videojogo através da nossa interface e de SQL Queries,, o que a torna "jogável". É possível matar Mobs e receber o item que estes *dropam*, realizar compras a Villagers, um tipo de mob, que vende itens em troca de uma esmeralda e também minerar/apanhar blocos do mundo.

Pretendemos que esta plataforma possa servir como uma "wiki" para jogadores que queiram pesquisar acerca das componentes do Minecraft para poderem melhorar as suas habilidades e poderem planear melhor as suas aventuras e, também, servir como suporte para criadores de *Mods* - conhecer as dinâmicas do jogo.

Para a realização deste projeto ambos os elementos do grupo trabalharam com o GitHub:

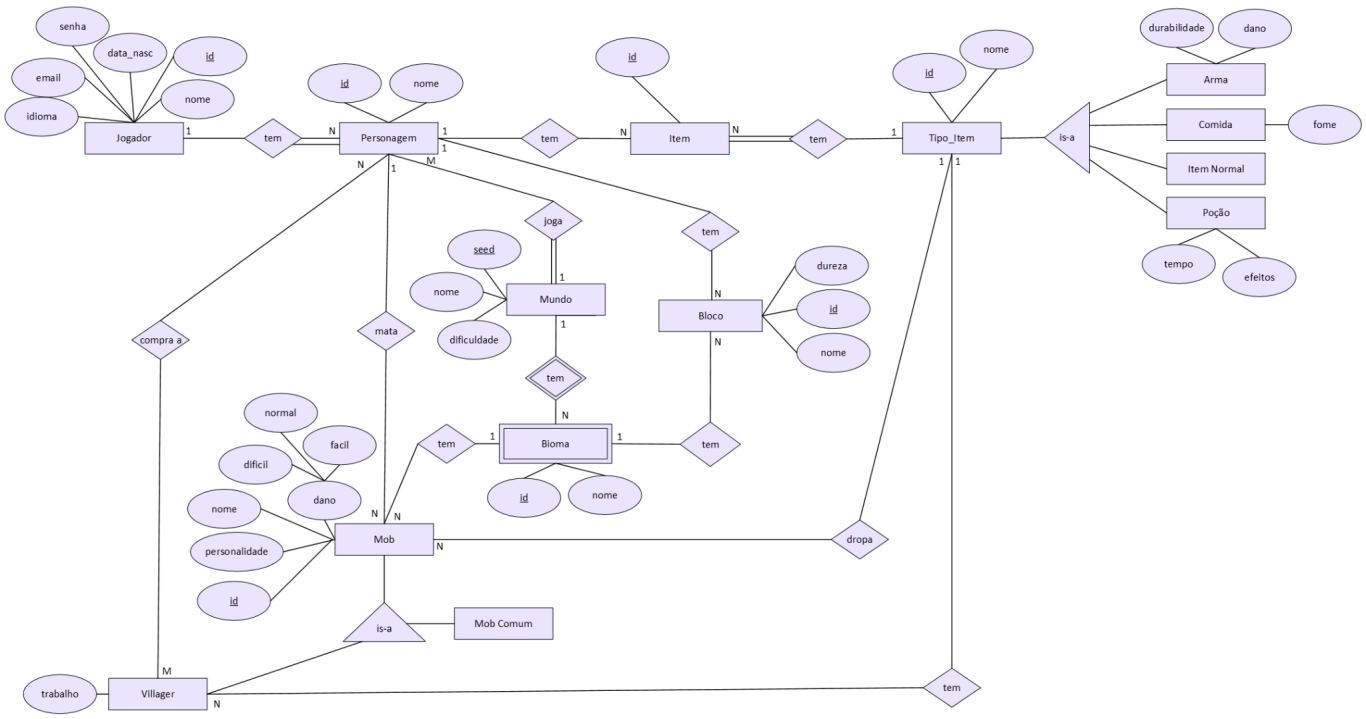
https://github.com/lilikas11/BD_MineBase

ANÁLISE DE REQUISITOS

- Um **Jogador** tem um ID, um Nome, um E-mail, uma Palavra-Passe, uma Data de Nascimento e um Idioma.
- Cada **Jogador** pode ter mais de uma **Personagem** que possui um ID e um Nome.
- Uma ou mais **Personagens** jogam num **Mundo** - único, é o mesmo para todos - que tem uma Seed, um Nome e uma Dificuldade.
- O **Mundo** é formado por vários **Biomas**, cada um caracterizado por um ID e um Nome.
- O **Bioma** possui **Mobs** e **Blocos.** Os **Mobs** possuem um ID, um Nome, uma Personalidade e um nível de dano que muda de acordo com a dificuldade do jogo (fácil, normal. difícil). Podem ser **Mobs Comuns** ou **Villagers**. Os **Villagers** têm um Trabalho e um **Tipo_ Item**. Os **Mobs** quando são mortos *dropam* um **Item** que passa a pertencer ao inventário da **Personagem**
- Um **Villager** pode vender **Itens** a um **Personagem**.
- Um **Bloco** tem um ID, um Nome e uma Dureza e pode pertencer a uma **Personagem** (estar no inventário da Personagem).
- Uma **Personagem** pode ter **Itens** no inventário, que possuem um ID. Um **Item** é uma instância da entidade **Tipo_Item** que possui um ID e um Nome. As **Armas** (Durabilidade e Dano), as **Comidas** (Fome), as **Poções** (Tempo e Efeitos) e os **Itens Normais** são tipos de **Tipo_Item**.

DER

Dado como terminada a fase de análise de requisitos, iniciámos a construção do nosso Diagrama de Entidades Relacional. Após várias iterações, chegámos ao seguinte diagrama:



ER

O seguinte esquema é a tradução do DER que se encontra na página anterior:

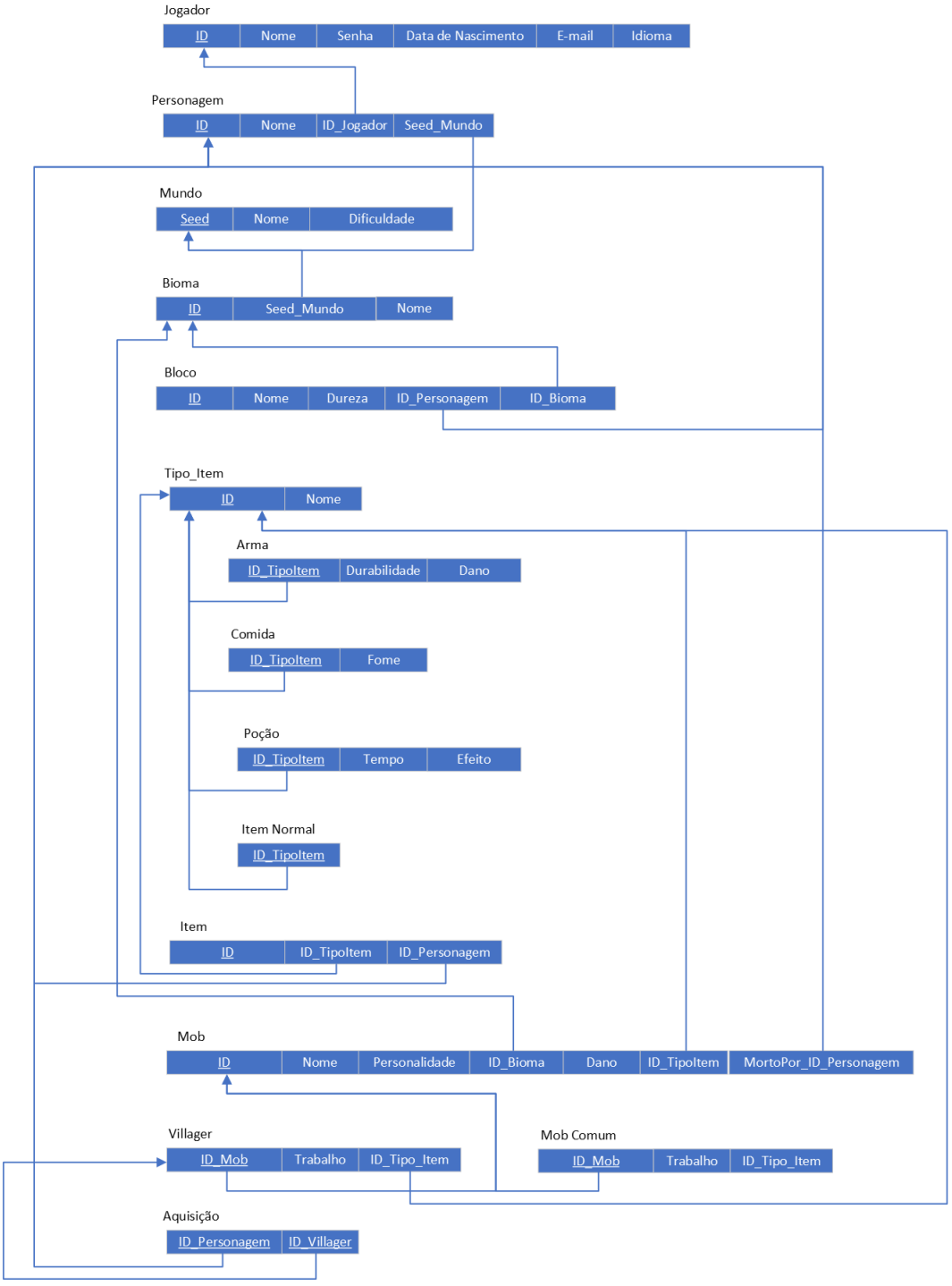
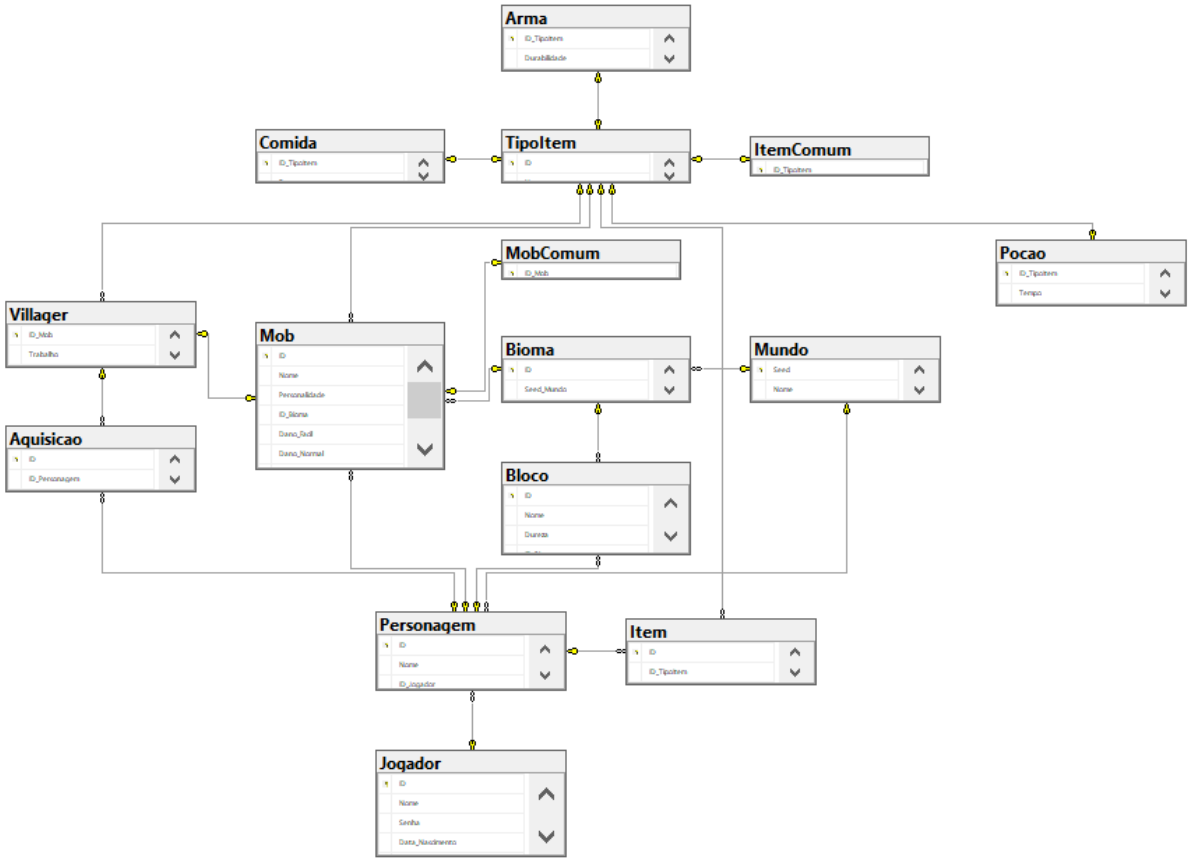


DIAGRAMA DA BD



INTERFACES

Para fazer a interface gráfica da MineBase utilizámos o Windows Forms e a linguagem de programação C#. Criámos apenas um Form denominado Form1.cs em que é possível:

- Realizar login na conta do Jogador. O utilizador apenas consegue entrar na MineBase se a combinação de E-mail e Palavra-Passe utilizada estiver registada na base de dados e se ambos os campos estão preenchidos - caso contrário são ativados pop-ups com mensagens de erro que pedem ao utilizador para realizar o login novamente usando dados válidos;
- Selecionar uma Personagem que pertence ao Jogador com que se fez login;

Após o processo de login, já no Mundo:

- Mudar o Bioma em que a Personagem se encontra. Os elementos presentes no Inventário do Mundo mudam de acordo com o Bioma selecionado;
- Adicionar Itens ao Inventário do Jogador através do botão “Add”, que abre um pop-up com uma lista de todos os itens disponíveis no jogo. O Jogador pode escolher qual pretende adicionar ao inventário;
- Filtrar os itens do Inventário do Jogador por “Tudo”, “Blocos”, “Itens”, “Armas”, “Comidas”, “Poções” e “Itens Comuns”;
- Filtrar os itens do Inventário do Mundo por “Tudo”, “Blocos”, “Mobs” e “Villagers;
- Consultar informações acerca dos elementos do jogo. Apenas aparecem preenchidos os campos que estão relacionados ao tipo de elemento selecionado (Bloco, Arma, Comida, Poção, Item Comum, Mob, Villager);
- Matar um Mob e receber no Inventário da Personagem o *drop* do mesmo;

- Minerar um Bloco e ficar com ele no Inventário do Jogador;
- Realizar trocas com Villagers. A Personagem pode trocar um item do tipo "esmeralda" por outro item característico do Villager com que realiza a troca, ficando com ele no seu Inventário;
- Passar ao próximo dia, ou seja, realizar um reset de todos os Mobs que foram mortos antes de se carregar no botão "Próximo Dia";
- Ativar a música do jogo através do botão "Som";

Existem funções que apesar de ainda não estarem implementadas na interface, já foram criadas funções em SQL para as implementar no futuro, como por exemplo: adicionar um novo Jogador, adicionar uma nova Personagem, criar uma *Kill Count* e realizar uma pesquisa de elementos do Mundo através de um mecanismo de análise de substrings.

STORED PROCEDURES

create_villager_with_mob:

Uma vez que os villagers são também mobs, queremos que na criação de um villager (insert de um value na tabela villager), seja também criado um mob. No entanto, surge a impossibilidade de fazermos esta ação a partir de um trigger, uma vez que os valores variáveis dos villagers são o valor do id do bioma (atributo do mob), o trabalho e o id do tipo de item (atributos de um villager). Para resolver este problema, fizemos então este procedure que recebe como inputs o id do bioma, o trabalho e o item de troca do villager, e faz então, insert dos valores default, juntamente aos valores dados, do villager e do mob.

EfetuaCompra:

Este procedure é usado quando o personagem compra algo ao villager. Nele é retirado uma esmeralda ao personagem, adicionado o item comprado ao inventário do mesmo, e adicionada a troca à tabela de trocas. Para prevenir o acontecimento de erros (exemplo: que seja retirado uma esmeralda ao personagem, mas não seja adicionado o item, por algum erro na base de dados) usamos uma transaction.

Matar:

No minecraft é possível matar mobs, para isto, é preciso ter uma arma e quando morto o mob dropa um item específico (definido como um atributo do mob).
Para representar esta ação, usamos este procedure que irá dar update no id do bioma para null, ao atributo morto por id do personagem para o id do personagem e adicionar o item correspondente ao drop do mob ao inventário do personagem.

Minerar:

É Também possível minerar um bloco, para tal usamos este procedure que dá update no Bioma do bloco para null, e coloca o id do personagem (atributo do bloco) como sendo o id do personagem que minerou o mesmo.

Passar o dia:

Quando se passa o dia no minecraft, novos mobs nascem no lugar onde o personagem se encontra. Para representar esta dinâmica fizemos então um procedure que utiliza de um cursor para percorrer todas as linhas da tabela mobs, e caso o id do bioma dos mesmos seja nulo (ou seja, caso esteja morto) dá update para o id do bioma onde o personagem se encontra.

AddJogador:

Adiciona um jogador recebendo o nome, senha, data, email e idioma.

AddPersonagem:

Adiciona um personagem recebendo o nome, o id do jogador e a seed do mundo.

Exemplos:

```
CREATE PROCEDURE EfetuaCompra(@id_personagem int, @id_villager int)
AS
BEGIN
        BEGIN TRANSACTION;
        BEGIN TRY
    -- Remove a esmeralda
    DELETE FROM Item WHERE ID = (SELECT TOP 1 ID FROM Item WHERE ID_Personagem = @id_personagem AND
ID_TipoItem = 81);

        Insert into Aquisicao values
               (@id_personagem, @id_villager)

    -- tipo item villager
    DECLARE @tipoItem int;
    SET @tipoItem = (SELECT ID_TipoItem FROM Villager WHERE ID_Mob = @id_villager);

    INSERT INTO Item VALUES (@tipoItem, @id_personagem);
               COMMIT;
           END TRY
           BEGIN CATCH
               ROLLBACK;
               THROW;
           END CATCH;
END
GO

-- PassarDia
CREATE procedure PassarDia(@id_bioma int)
```

```
AS
BEGIN
    declare @mob_id INT;

    declare mob_cursor CURSOR FOR
    SELECT ID
    FROM Mob
    WHERE ID_Bioma IS NULL;

    OPEN mob_cursor;

    fetch NEXT FROM mob_cursor INTO @mob_id;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        UPDATE Mob
        SET ID_Bioma = @id_bioma,
            MortoPor_ID_Personagem = NULL
        WHERE ID = @mob_id;

        fetch NEXT FROM mob_cursor into @mob_id;
    END;

    close mob_cursor;
    deallocate mob_cursor;
END
go
```

UDF'S

Login:

Quando um Jogador dá login no sistema é necessário confirmar se o utilizador e a senha do mesmo existem no banco da base de dados. Esta função confirma então este parâmetros e retorna, caso exista, o ID do Jogador correspondente, caso não exista, retorna Null.

PodeComprar:

Pode comprar é uma função que retorna um valor de 0 ou 1 (0 = falso, 1 = true) e confirma se o jogador tem uma esmeralda para comprar um item ao villager.

Inventário:

A função Inventário retorna uma table usando a InventárioView onde o id do personagem é o personagem atual.

Bioma:

A função Bioma retorna uma table usando a MundoView onde o id do bioma é o bioma atual.

TemArma:

Para minerar ou matar um mob, o jogador precisa de ter uma arma na sua posse. Para isso usámos esta função que percorre o inventário do jogador, confere se o mesmo tem, ou não uma arma e retorna um valor de 0 ou 1 (0 = falso, 1 = true).

Kills:

Para disponibilizar ao player uma estatística sobre as kills que já fez durante o seu jogo, fizemos esta função que faz a conta dos mobs que foram mortos, num dia, pelo personagem.

ItemInfo, BlocoInfo e MobInfo:

Na nossa interface quisemos disponibilizar todas as informações sobre os Objetos do jogo, para tal criamos 3 funções que recebem o id do objeto correspondente e retornam informações importantes sobre os mesmos.

FiltroMundo:

De modo a facilitar a viagem do personagem pelo mundo, criamos um filtro onde o player pode escolher apenas ver Blocos, Mobs ou Villagers.

FiltroInventario:

Durante o jogo de minecraft, pode acontecer de o inventário ficar *overloaded* (com demasiados items), para facilitar a ação do player de confirmar se têm um item específico, criamos um filtro com os seguintes atributos: Bloco, Item, Arma, Comida, Poção, Item Comum.

SearchMundo:

Numa versão de alta carga, o mundo pode ter imensos blocos e imensos mobs. Considerando esta circunstância, implementamos também uma função que recebe uma string (dada pelo player) e retorna as linhas da base de dados do Mundo cujo o nome ou tipo contenha a string dada.

Exemplo:

```
-- Função para ver se o player tem arma
create function TemArma(@id_personagem int) returns int
as
        begin
        declare @status int
        if(select top 1 I.ID
                from Item I
                Join TipoItem T on T.ID = I.ID_TipoItem
                Join Arma A on A.ID_TipoItem = T.ID) is not null
        BEGIN
         SET @status = 1;
    END
    ELSE
    BEGIN
        SET @status = 0;
    END

    RETURN @status
```

```
end
go


-- count de kills
create function kills(@id_personagem int)
returns int
as
begin
	return(
		select COUNT(*) as KillsCount
			from Mob where MortoPor_ID_Personagem = @id_personagem
	)
	end
go
```

TRIGGERS

Delete_Jogador, Delete_Mundo, Delete_personagem, Delete_Bioma, Delete_TipoItem, Delete_Mob, Delete_Bloco, Delete_Aquisicao, Delete_Villager:

Estes triggers são acionados quando algum item é apagado ou atualizado, de maneira a que as entradas das tabelas dependentes sejam também atualizadas.

CheckEmailJogador e CheckIdioma:

Quando adicionado um novo jogador são acionados estes triggers que vêm respetivamente se o email do jogador está na forma de email e se o idioma está dentro dos idiomas listados na nossa base de dados,

Exemplos:

```
CREATE TRIGGER Delete_Mundo ON Mundo
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;
    DELETE FROM Personagem WHERE Seed_Mundo IN (SELECT Seed FROM deleted);
    DELETE FROM Mundo WHERE Seed IN (SELECT Seed FROM deleted);
END
GO
```

```
-- Loggin
create trigger checkEmailJogador on Jogador for insert, update
as
	begin
		if(UPDATE(Email) and exists(select 1 from inserted where not Email like '%_@__%.__%')
)
		begin
			raiserror('Insira um email válido', 16, 1);
			ROLLBACK;
		end
	end
go
```

VIEWS

InventárioView:

O Inventário de um jogador é constituído por Blocos e Items, por isso, para reunirmos estes objetos e apresentá-los em conjunto numa lista, usamos a mundo view.

MundoView:

O mundo de Minecraft é constituido por Blocos e mobs, à semelhança do inventário, usámos uma view para juntar tudo em uma única lista.

Exemplos:

```
 -- mundo view
Create view MundoView AS

	select B.ID, B.Nome, 'Bloco' as Tipo, B.ID_Bioma
	from Bloco B
	where B.ID_Bioma is not null

	union all

	select M.ID, M.Nome,
		case when V.ID_Mob is not null
			then 'Villager'
			else 'Mob'
			end as Tipo,
		M.ID_Bioma
	from Mob M
	left join Villager V on M.ID = V.ID_Mob
	where M.ID_Bioma is not null

go
```

INDEXES

InventárioView:

O único index que vimos justificado usar foi uma referência ao Nome dos Biomas, para conseguirmos organizar os mesmos por ordem alfabética

Exemplo:

```
create index idx_Nome
on Bioma(Nome)
```

CONCLUSÃO

Este projeto foi uma excelente oportunidade para pormos em prática os conhecimentos que adquirimos ao longo do semestre acerca da criação e das dinâmicas de bases de dados. Também pudemos estudar melhor a interação entre as bases de dados e as interfaces que as utilizam, uma vez que este projeto nos requisitava uma interface intuitiva.
Pensamos que atingimos com sucesso os objetivos demarcados para este projeto e que fomos capazes de trabalhar com as ferramentas disponibilizadas.

BIBLIOGRAFIA

Para realizar este projeto foram consultados os PowerPoints disponibilizados para as aulas teóricas e os guiões disponibilizados para as aulas práticas.
Além disso, foi consultada a documentação de Windows Forms:

- Microsoft. "Windows Forms documentation". Microsoft Learn. Disponível em: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-7.0. Consultado a 3 de junho de 2023.