SENAI-SP

UC13

Atividade online 1
Documentação Preliminar
Projeto para Banco de Dados
JOGO RPG Online

Aluno: Wesley B. Carvalho
Data: Jan/2023

OBJETIVOS DA ATIVIDADE

De acordo com a atividade proposta, ficou estabelecido para o aluno desenvolver uma documentação técnica associada ao projeto de banco de dados RPG online que contemplasse os seguintes tópicos:

- Escopo
- Requisitos
- Modelagem conceitual (MER/DER)

Tópicos complementares adicionados pelo aluno:

- Modelo Lógico
- Dicionário de Dados
- Modelo Físico

Índice de figuras

Sumário

1. ESCOPO

É importante situar-se no domínio do problema, pois ele delimita o escopo do projeto de banco de dados. Ele estabelece a fronteira do que deve estar no banco de dados.

Independente da complexidade do domínio do problema, todo domínio possui requisitos de informação e regras de negócio que devem ser identificados e interpretados pelo analista de sistemas a fim de serem corretamente representadas no modelo conceitual de dados.

Neste jogo RPG, o domínio será constituído pelos dados dos usuários, personagens, classes e habilidades que compõem o mundo virtual.

Deve-se ressaltar que estão previstas atualizações no teor desta documentação, uma vez que trata-se de *algo preliminar em análise* até que seja estabelecida a documentação principal do projeto a qual será mais detalhada.

1.1. OBJETIVOS

- Criação de um jogo online de modo que possa atender a usuários de todas as faixas etárias que se identifiquem ou tenham paixão por jogos RPG.
- Atingir um maior número de pessoas, proporcionando entretenimento, conhecimento e amadurecimento em relação a traçar estratégias adequadas dentro do cenário do jogo. Porém, para um jogo digital funcionar corretamente, se faz necessária a manipulação e o armazenamento de grande quantidade de dados, que serão gerenciados pela criação de um banco de dados.

1.2. RECURSOS

1.2.1. AMBIENTE DE TRABALHO

O produto será desenvolvido em ambiente próprio e adequado, tendo computadores com Sistema Operacional Windows 10 em rede local e com acesso à internet. Também será requisitado um servidor para execução do projeto.

1.2.2. MÃO DE OBRA

Um funcionário habilitado e com conhecimento técnico em banco de dados pertencente a uma equipe, irá desenvolver o projeto.

1.2.3. FERRAMENTAS

Para uso no projeto serão utilizados:

- SQLServer 2019;
- Ferramentas SGBD (Sistema de Gerenciamento de Banco de Dados) como sendo o SSMS (SQL Server Management Studio, versão 18.11.1);
- Astah, Lucidchart e/ou Drawio;
- Dbdocs.io;
- Pacote Office para a criação de relatórios ou documentações específicas.

1.3. ORÇAMENTO

Estima-se que os custos que envolvam a elaboração deste projeto ultrapassem o valor de R$ 10.000,00, considerando os recursos envolvidos, mão de obra especializada e encargos.

Uma planilha de custos mais detalhada e assertiva será disponibilizada quando na atualização desta documentação preliminar.

2. REQUISITOS

São obtidos durante o levantamento de requisitos para análise junto à empresa contratante especializada em jogos, para poder compreender o domínio da aplicação e a definição de prioridades. Os requisitos devem ser verificados para avaliar se estão completos e consistentes dentro do planejamento.

2.1. FUNCIONAIS

Representam as funções ou serviços que o sistema deve realizar de acordo com as necessidades do cliente. Inicialmente foi estabelecido que um usuário poderá:

- Realizar o cadastro no site.
- Fazer o login no site.
- Criar/selecionar classe.
- Criar habilidade.
- Criar personagem.

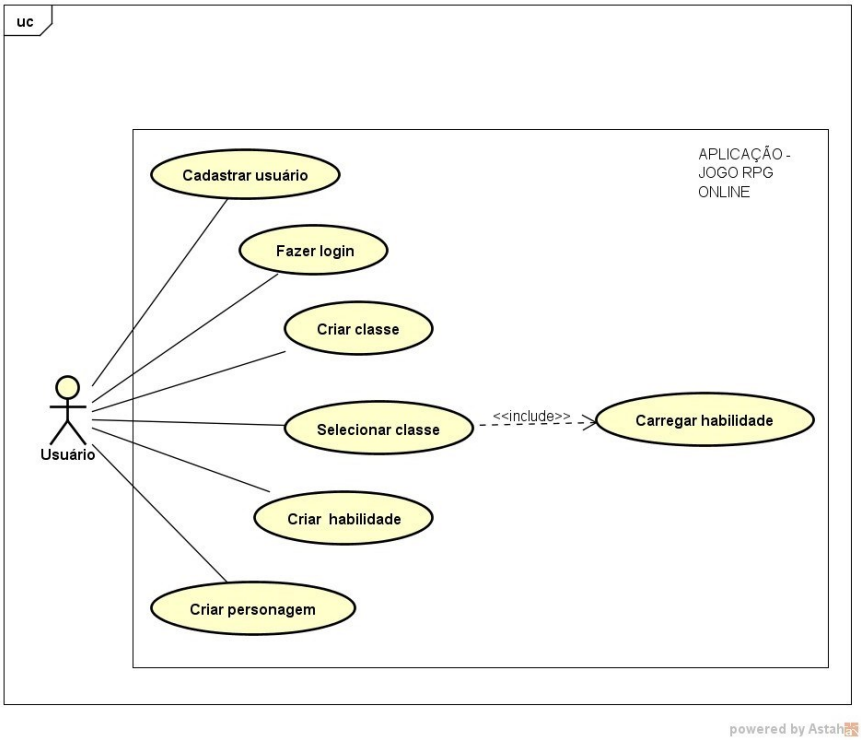

Figura 1: Diagrama de Caso de Uso

2.2. NÃO-FUNCIONAIS

Representam os requisitos de qualidade, como sendo os requisitos que especificam critérios que podem ser usados para descrever o funcionamento de um sistema. Neste projeto foram descritos alguns aspectos gerais iniciais.

- Compatibilidade: Por ser uma aplicação web, será necessário o uso de um navegador atualizado (Chrome, Firefox ou Safari), podendo ser acessada de um computador (Desktop/Laptop), tablet ou celular que disponha de tal recurso e acesso à internet, independente do Sistema Operacional.
- Desempenho
  - *Tempo de resposta*: Deverá ser de no máximo 2s.
- Segurança: Os dados do usuário deverão ser armazenados e mantidos seguros no banco de dados SQL Server, não podendo sofrer nenhum tipo de interferência ou vazamento de dados para terceiros, sendo que somente devem ser disponibilizados a usuários devidamente autorizados pelo sistema com o uso de login e senha durante a autenticação. As senhas dos usuários devem ser criptografadas.

3. MODELO CONCEITUAL(MER/DER)

3.1. MER (MODELO ENTIDADE-RELACIONAMENTO)

Nesta seção, será abordado *o Modelo Entidade-Relacionamento (MER).*
***Para o jogo RPG online**:*

Um usuário deverá acessar a plataforma de jogo através de seu email e senha. Caso seja seu primeiro acesso, será necessário realizar o cadastro no sistema informando estes dados. Um usuário apenas poderá ter um personagem exclusivo no jogo, que terá nome e classe específica.

Cada Personagem pertence a uma classe, que possui nome e descrição, considerando que cada classe possui muitos personagens. Uma classe poderá ter muitas habilidades, onde cada uma delas será identificada por um nome, de forma que uma habilidade em comum também pode ser vista em muitas classes.

De forma geral, cada usuário possui apenas um personagem, relacionado a uma classe e que tem uma ou mais habilidades.

Posteriormente será feito o DER contendo os diagramas. Tendo como base as regras de negócio e o levantamento de requisitos, foram identificadas as entidades e os atributos pertinentes a cada entidade, conforme Figura 2.

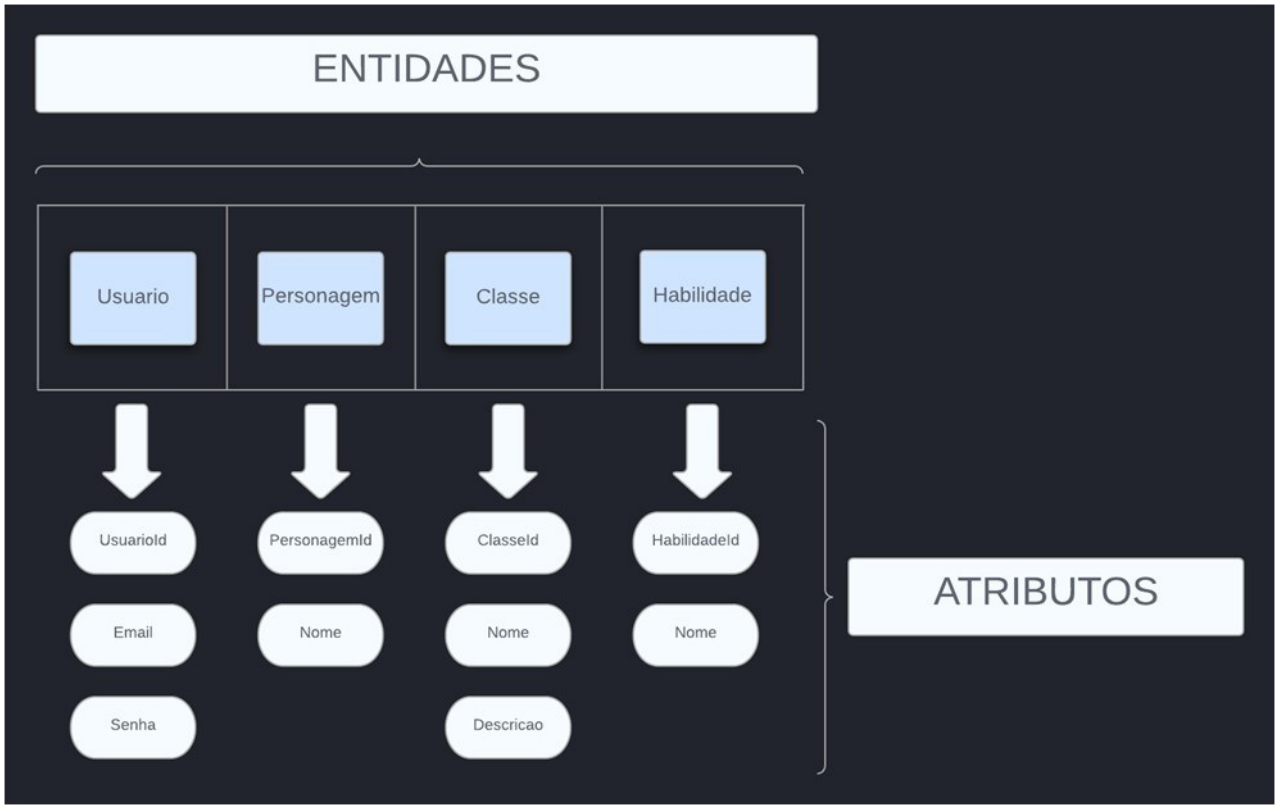

Figura 2: Entidades e atributos

3.2. DER (DIAGRAMA ENTIDADE-RELACIONAMENTO)

Foi criado um diagrama inicial representando o modelo conceitual para identificar as entidades, atributos, relacionamentos com as cardinalidades (grau do relacionamento) a fim de obter uma visão geral do projeto para o jogo RPG (Vide Figura 3). Os atributos em amarelo, representam os atributos identificadores de cada entidade.

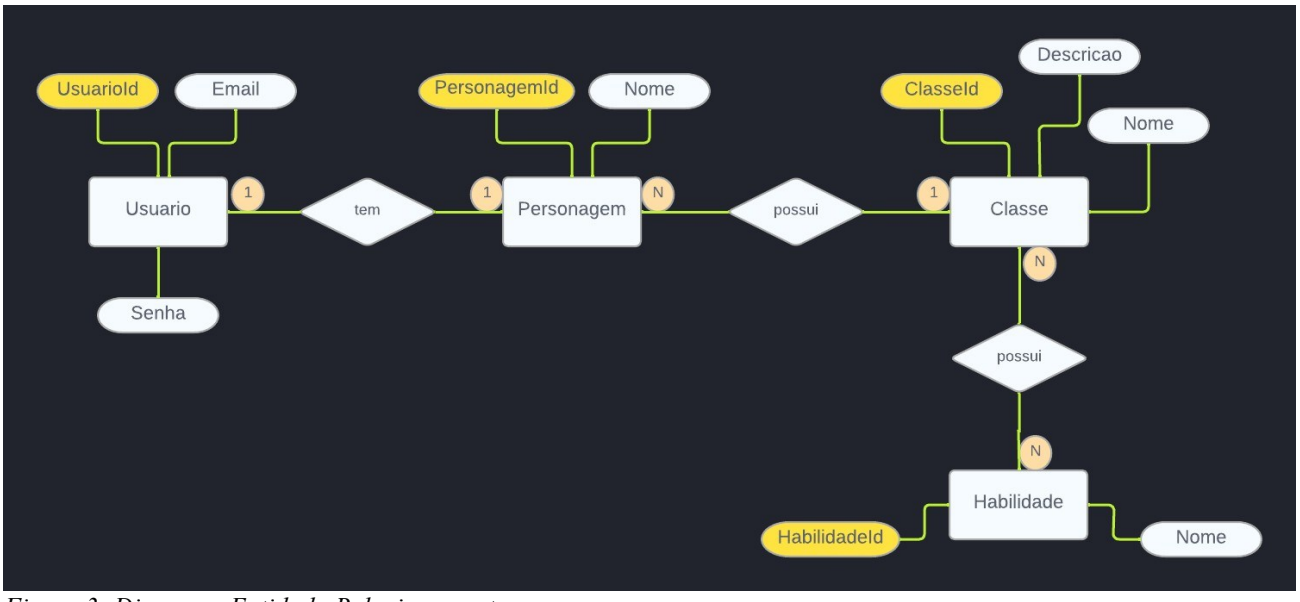

Figura 3: Diagrama Entidade-Relacionamento

4. MODELO LÓGICO

Em seguida foi desenvolvido o modelo lógico, por meio de um diagrama onde as entidades passaram a ser representadas por tabelas e os atributos por campos (Vide Figura 4). Observe que nos relacionamentos, a cardinalidade está sendo demonstrada por (1:1 – um para um) e (1:* - um para muitos).

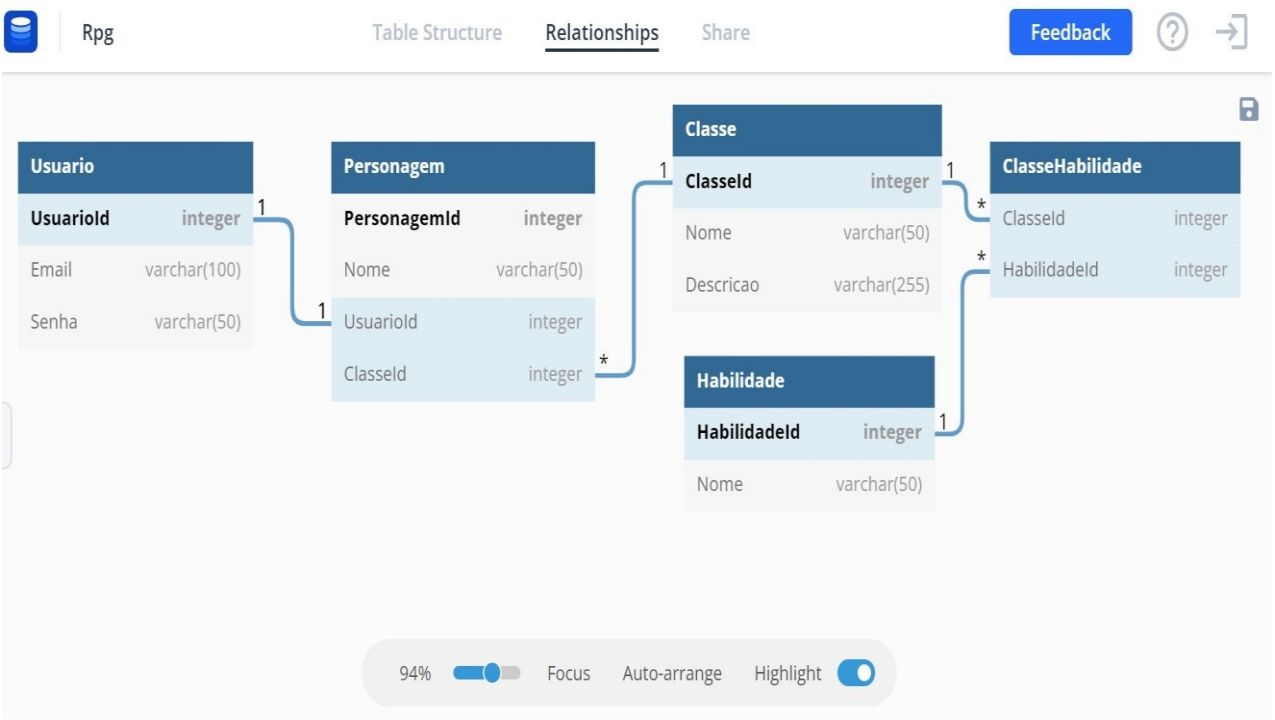

Figura 4: Tabelas e Relacionamentos usando Dbdocs.io

5. DICIONÁRIO DE DADOS

Para melhor descrever, organizar, documentar e ter um controle sobre os dados do modelo lógico do projeto, foi criado um dicionário de dados, contendo os nomes das tabelas, campos, o tipo de dado, configuração para os campos, e os relacionamentos indicando a cardinalidade. Em cada campo também foi adicionada uma descrição. O dicionário de dados mostra a estrutura de armazenamento de cada tabela.

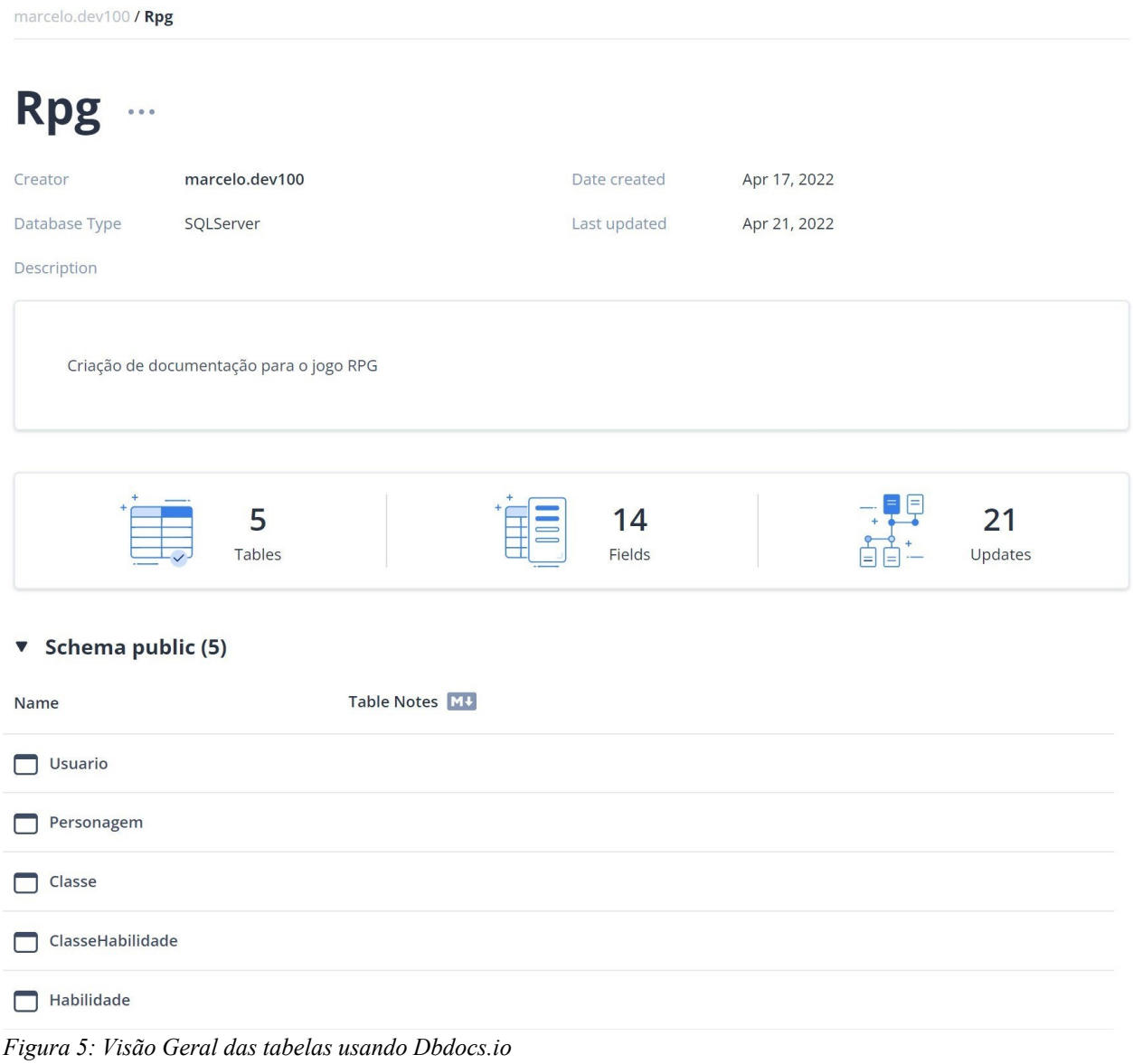

Figura 5: Visão Geral das tabelas usando Dbdocs.io

marcelo.dev100 / Rpg / public / Usuario

public.Usuario

Creator marcelo.dev100

Notes

Fields (3)

Name	Type	Settings	References	Default Value & Notes
UsuarioId	integer	PK, increment	Personagem.UsuarioId	Identificador único de um registro na tabela Usuário
Email	varchar(100)	not_null, unique		Usado como login do usuário
Senha	varchar(50)	not_null		Senha para cadastro de um usuário

Figura 6: Dicionário de dados - Tabela Usuario

marcelo.dev100 / Rpg / public / Personagem

public.Personagem

Creator marcelo.dev100

Notes

Fields (4)

Name	Type	Settings	References	Default Value & Notes
PersonagemId	integer	PK, increment		Identificador único de um registro na tabela Personagem
Nome	varchar(50)	not_null, unique		Nome de um personagem
UsuarioId	integer	unique	Usuario.UsuarioId	Chave estrangeira que faz referência à chave primária UsuarioId da tabela Usuario
ClasseId	integer		Classe.ClasseId	Chave estrangeira que faz referência à chave primária ClasseId da tabela Classe

Figura 7: Dicionário de dados - Tabela Personagem

marcelo.dev100 / Rpg / public / Classe

public.Classe

Creator marcelo.dev100

Notes

Fields (3)

Name	Type	Settings	References	Default Value & Notes
ClasseId	integer	PK, increment	Personagem.ClasseId ClasseHabilidade.ClasseId	Identificador único de um registro na tabela Classe
Nome	varchar(50)	not_null, unique		Nome de uma classe específica
Descricao	varchar(255)			Descreve uma classe com detalhamento

Figura 8: Dicionário de dados - Tabela Classe

marcelo.dev100 / Rpg / public / ClasseHabilidade

public.ClasseHabilidade

Creator marcelo.dev100

Notes

Fields (2)

Name	Type	Settings	References	Default Value & Notes
ClasseId	integer		Classe.ClasseId	Chave estrangeira que faz referência à chave primária ClasseId da tabela Classe
HabilidadeId	integer		Habilidade.HabilidadeId	Chave estrangeira que faz referência à chave primária HabilidadeId da tabela Habilidade

Figura 9: Dicionário de dados - Tabela ClasseHabilidade

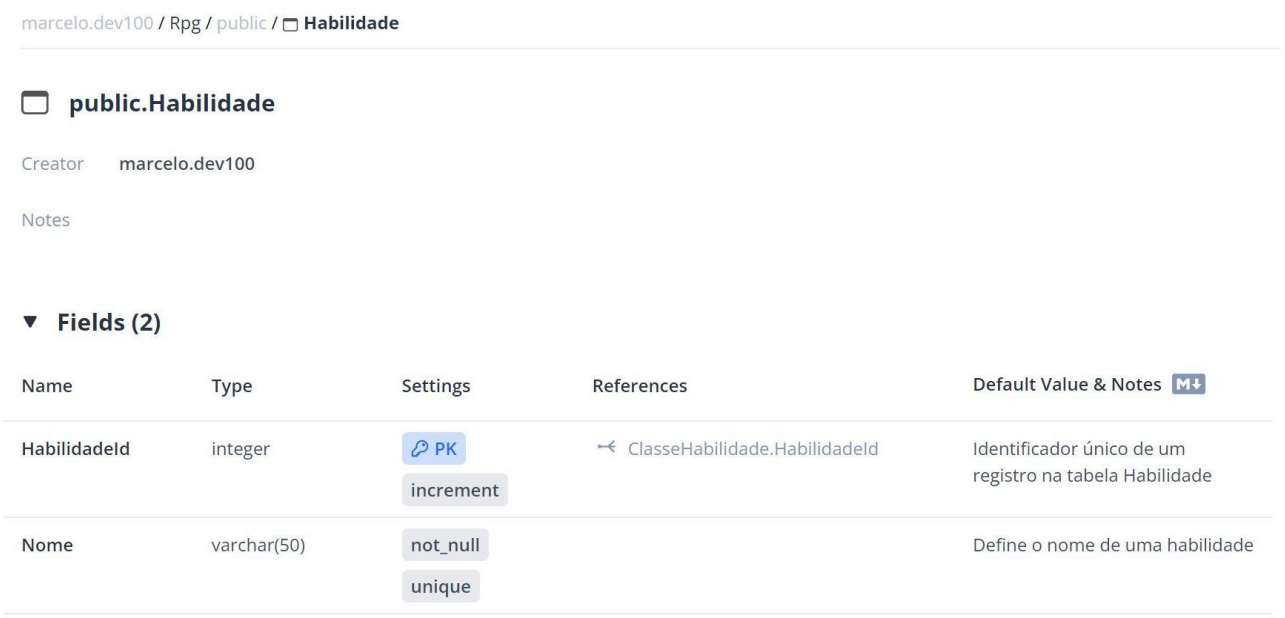

Figura 10: Dicionário de dados - Tabela Habilidade

6. MODELO FÍSICO

Depois de ter estruturado o modelo conceitual (MER/DER) e convertido para o modelo lógico (modelo relacional/banco de dados relacional), é necessário iniciar a implementação do modelo físico. Esta etapa descreve o modo de armazenamento no banco por meio de uma linguagem de programação. Neste caso será usado o SQL (Linguagem de Consulta Estruturada).

6.1. DDL (Data Definition Language)

Consiste na Linguagem de Definição de Dados. É usada para definir ou modificar estrutura de dados. É usada a sintaxe SQL para descrever os dados. Neste projeto está sendo utilizada para criação do banco de dados RPG e as tabelas com as respectivas colunas, tipos de dados, entre outros. Antes da criação das tabelas, é necessário alternar para o banco de dados que será utilizado, pelo comando USE RPG.

```sql
--DDL
CREATE DATABASE RPG
GO

USE RPG
GO

CREATE TABLE Usuario
(
  UsuarioId INT PRIMARY KEY IDENTITY,
  Email VARCHAR(100) UNIQUE NOT NULL,
  Senha VARCHAR(50) NOT NULL
)
GO

CREATE TABLE Classe
(
  ClasseId INT PRIMARY KEY IDENTITY,
  Nome VARCHAR(50) UNIQUE NOT NULL,
  Descricao VARCHAR(255)
)
GO

CREATE TABLE Personagem
(
  PersonagemId INT PRIMARY KEY IDENTITY,
  Nome VARCHAR(50) UNIQUE NOT NULL,
  UsuarioId INT UNIQUE FOREIGN KEY REFERENCES Usuario(UsuarioId),
  ClasseId INT FOREIGN KEY REFERENCES Classe(ClasseId)
)
GO

CREATE TABLE Habilidade
(
  HabilidadeId INT PRIMARY KEY IDENTITY,
  Nome VARCHAR(50) UNIQUE NOT NULL
)
GO

CREATE TABLE ClasseHabilidade
(
  ClasseId INT FOREIGN KEY REFERENCES Classe(ClasseId),
  HabilidadeId INT FOREIGN KEY REFERENCES Habilidade(HabilidadeId)
)
```

Figura 11: Comandos para criação de tabelas

6.2. DML (Data Manipulation Language)

Consiste na Linguagem de Manipulação de Dados. Fornece um conjunto de comandos para realizar as manipulações dos dados armazenados (INSERT, UPDATE, DELETE). Em algumas literaturas, o comando SELECT também é incluído como DML. Neste projeto está sendo usada para inserção de dados na tabela.

```
--DML
--adicionando dados

USE RPG
GO

INSERT INTO Usuario VALUES ('email@email.com', 123456)
INSERT INTO Classe VALUES ('Bárbaro' , 'Descrição do Bárbaro')
INSERT INTO Habilidade VALUES ('Lança Mortal'), ('Escudo Supremo')
INSERT INTO Personagem (Nome, UsuarioId, ClasseId) VALUES ('DeuBug', 1, 1)
INSERT INTO ClasseHabilidade(ClasseId, HabilidadeId) VALUES (1, 1), (1, 2)


INSERT INTO Usuario VALUES ('outroemail@email.com', 123456)
INSERT INTO Classe VALUES ('Monge' , 'Descrição do Monge')
INSERT INTO Habilidade VALUES ('Recuperar Vida')
INSERT INTO Personagem (Nome, UsuarioId, ClasseId) VALUES ('BitBug', 2, 2)
INSERT INTO ClasseHabilidade(ClasseId, HabilidadeId) VALUES (2, 2), (2, 3)
```

Figura 12: Comandos para inserção de dados nas tabelas

Em tempo: Nesta parte, para fins de estudo, a senha é gravada na sua forma original. Porém, não é recomendado, já que deve ser criptografada.

6.3. DQL (Data Query Language)

Consiste na Linguagem de Consulta de Dados. Permite recuperar os dados através do comando SELECT de uma tabela ou de tabelas relacionadas, sendo que neste último caso, são usadas cláusulas associadas.

```
--DQL
SELECT * FROM Usuario
SELECT * FROM Personagem
SELECT * FROM Classe
SELECT * FROM Habilidade
SELECT * FROM ClasseHabilidade
```

Figura 13: Comandos para consulta de dados nas tabelas