Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

"Will It Run?"

Guilherme Pereira Martins (A70782)
Rui Alves dos Santos (A67656)
André Rodrigues Soares (A67654)
Paulo Henrique Alves (A64282)

Novembro, 2018

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

“Will It Run?”

Guilherme Pereira Martins (A70782)
Rui Alves dos Santos (A67656)
André Rodrigues Soares (A67654)
Paulo Henrique Alves (A64282)

Novembro, 2018

Resumo

Com este projeto pretendemos criar uma base de dados relacional, desde a análise e modelação, até a sua implementação física e utilização num SGBD (sistema de gestão de base de dados). Este projeto, intitulado de “Will it run” baseia-se num já existente chamado “Can You Run It” que permite saber se determinados jogos correm num determinado hardware específico.

O presente projeto visa criar uma base de dados que permita guardar informação de conjuntos de hardware, CPU, GPU e RAM e agrupá-los em configurações que um utilizador possui e também dar resposta aos utilizadores de forma rápida e eficiente sobre que jogos podem utilizar nessas configurações.

Tal base de dados poderá posteriormente ser utilizada por utilizadores para não só saber a qualquer momento que jogos são compatíveis ou não com o seu hardware, mas também dar suporte para quando quiserem fazer trocas de hardware para correr de forma eficiente jogos antigos ou novos, porém este projeto só trata da parte da criação e gestão do sistema de base de dados.

Área de Aplicação: Desenho, arquitetura e administração de base de dados para o serviço “Will it run?”

Palavras-Chave: Hardware, CPU, GPU, RAM, Requisitos, Jogo, SGBD, Modelo relacional, transação

Índice

Índice de Figuras

Índice de Tabelas

1. Definição do Sistema

1.1. Contexto de aplicação do Sistema

A indústria dos videojogos move milhares de milhões de euros por ano e foi uma das que mostrou um crescimento mais significativo nas últimas décadas empregando milhões de pessoas atualmente e adicionando mais jogadores à comunidade diariamente. A constante evolução na computação permite que todos os anos sejam lançados novos jogos com melhorias significativas relativamente ao ano anterior. Este crescimento na indústria é proporcional à evolução na qualidade do hardware sendo o mercado invadido constantemente por novos computadores mais potentes. No caso de jogadores de computador é necessário saber se a sua máquina tem capacidade para correr os novos jogos ou mesmo os mais antigos com maior fidelidade gráfica, que tem associados requisitos mínimos e recomendados, disponibilizados pelos próprios desenvolvedores dos jogos.

"Posso correr o jogo X com o computador que tenho? e o jogo Y? Tenho que comprar um componente melhor?", são perguntas que um consumidor tem que fazer antes de assumir a compra de um determinado jogo. Esta avaliação e comparação do nosso hardware com o que o jogo pede pode por vezes ser complicada para um utilizador casual.

Foi neste âmbito que nasceu o "Will it run", baseado no "Can You RUN it", um website que faz exatamente isso, um scan do hardware presente no nosso computador e depois diz se corre jogo X ou Y, se só é suficiente para correr os jogos nas configurações mínimas ou se é recomendado e permite uma maior fidelidade gráfica. Porém é sempre necessário que seja executado esse "scanner" disponibilizado e depois enviado para o website, foi por essa razão que decidimos criar uma alternativa onde um utilizador vai simplesmente ao website, escolhe os componentes que tem fazendo assim uma configuração e depois compara com os requisitos dos jogos. Cada componente vai ser avaliado e comparado com um mínimo listado.

1.2. Apresentação do Caso de Estudo

Jogos, GPUs, CPUs e RAM são componentes constante atualização, a melhor placa gráfica hoje poderá estar desatualizada amanhã, a minha memória RAM de 8GB pode não ser suficiente para ter o sistema operativo com vários processos e um jogo a correr ao mesmo tempo, há um constante despejo de informação e de novo hardware/software no mercado que é difícil de acompanhar. O nosso caso de estudo vai ser o "Will it Run" um serviço que tem como objetivo catalogar os diversos componentes de hardware, bem como o seu "poder de execução", entre outras informações e os requisitos dos jogos em questão. Por fim catalogar a

relação de “poder” entre cada componente e jogo, visto que nem tudo executa linearmente. O site permite aos utilizadores uma utilização simples como saber se o seu computador será capaz de correr um jogo ou escolher e guardar uma configuração com componentes previamente inseridos na base de dados. O administrador do “Will it run” pretende adotar um sistema de gestão de base de dados que permita uma melhor organização e manutenção da informação do hardware e dos jogos de forma a estar sempre atualizado.

1.3. Análise da viabilidade do processo

No que diz respeito à informática, há um grande intervalo de problemas que podem surgir quando estamos a instalar um novo jogo, particularmente com um que acabou de sair no mercado. Hardware não suportado ou em falta ou um sistema operativo incorreto podem ser a causa de múltiplas dores de cabeça e infinitas horas de suporte técnico e em último caso a devolução do jogo ou gastos extras para trocar componentes, sem falar quando essa troca não é bem feita e há incompatibilidades de hardware ou hardware que vem defeituoso de fábrica. Após a montagem do seu computador ou compra de um, uma pessoa procura sempre poder jogar os videogames mais recentes que lhe despertam a atenção, muitas vezes dão por si a comprar um jogo e a descobrir que o seu computador não tem capacidade para o executar. Com o serviço “Will it run?” pretendemos fornecer a essa pessoa o limite estabelecido pelo seu hardware e que videojogos pode correr na sua máquina sem ultrapassar esse limite, podendo assim usufruir da capacidade que o seu hardware lhe fornece.

Em relação à análise e criação de base de dados, visto que tanto as informações dos componentes de hardware e os requisitos para os jogos está sempre disponível online o processo de criação e exploração dessa informação para a “montagem” do “Will it run” não apresenta qualquer problema.

2. Levantamento e Análise de Requisitos

2.1. Método de levantamento e análise de requisitos adotados

Uma das etapas mais importantes para o correto funcionamento de uma base de dados é o levantamento e respetiva análise dos requisitos.
Nesta fase não só identificamos os vários requisitos de qual informação queremos que o nosso sistema suporte, mas também como essa mesma informação está organizada e interage entre si.

Vale lembrar que não é só importante fazer uma análise atual, mas também uma de como o sistema deverá se comportar para aplicações futuras e tendo em conta o seu crescimento ano após ano.

Como temos já temos uma base de como a informação deve ser guardada e organizada, sendo essa base o website “Can You RUN it”, a nossa metodologia foi de observar nesse próprio website as funcionalidades e retirar daí os nossos requisitos. Numa segunda fase fizemos no próprio grupo algo parecido a uma análise “cliente - analista” e entre nós verificamos e retiramos requisitos adicionais que poderiam não ter sido descobertos na primeira análise do website.

Tendo também em conta que não há requisitos específicos para diferentes tipos de utilizadores optamos por usar um modelo centralizado onde todos os requisitos estão descritos em conjunto.

2.2. Requisitos Levantados

Os requisitos foram posteriormente organizados e divididos em grupos. Tendo feito isto podemos ver com melhor clareza os requisitos que já tínhamos levantado e fazer uma última análise para melhor descrever estes requisitos e ocasionalmente até encontrar algum que tivesse em falta, mas que faria sentido estar contemplado no nosso sistema.

2.2.1. Requisitos de descrição

Os seguintes requisitos identificam o esquema da base de dados e as suas possíveis vistas de utilização.

1. A base de dados deve permitir inserir informação sobre processadores (CPU)
    a. A base de dados deve permitir inserir o modelo de CPU
    b. A base de dados deve permitir inserir a marca do CPU
    c. A base de dados deve permitir inserir o ano em que foi lançado um CPU
    d. A base de dados deve permitir inserir o rating (pontuação relativa) de um CPU
    e. A base de dados deve permitir inserir a velocidade do CPU, ex:2.4GHz
2. A base de dados deve permitir inserir informação sobre processadores gráficos (GPU)
    a. A base de dados deve permitir inserir o nome ou modelo de um GPU
    b. A base de dados deve permitir inserir a marca do GPU
    c. A base de dados deve permitir inserir o ano em que foi lançado um GPU
    d. A base de dados deve permitir inserir o rating (pontuação relativa) de um GPU
    e. A base de dados deve permitir inserir a velocidade do GPU, ex: 2GHz
    f. A base de dados deve permitir inserir a quantidade de memória de um GPU (VRAM).
3. A base de dados deve permitir inserir informação sobre memórias RAM
    a. A base de dados deve permitir inserir o nome da memória RAM
    b. A base de dados deve permitir inserir a marca da memória RAM
    c. A base de dados deve permitir inserir o ano em que foi lançada a memória RAM
    d. A base de dados deve permitir inserir a capacidade da memória RAM
    e. A base de dados deve permitir inserir a velocidade da memória RAM, ex:240 MHz
4. A base de dados deve permitir inserir informação sobre Utilizadores
    a. A base de dados deve permitir inserir o nome de um utilizador
    b. A base de dados deve permitir inserir informação sobre as configurações de hardware do sistema de um utilizador (CPU,GPU,RAM)
    c. A base de dados deve permitir inserir um nome para a configuração de hardware de um utilizador
    d. A base de dados deve permitir inserir em que data foi criada uma configuração de hardware de um dado utilizador
5. A base de dados deve permitir inserir informação sobre jogos
    a. A base de dados deve permitir inserir o nome do jogo
    b. A base de dados deve permitir inserir o ano em que lançado
    c. A base de dados deve permitir inserir o nome da desenvolvedora(s) que criaram um jogo
    d. A base de dados deve permitir inserir quais são os requisitos mínimos e máximos de CPU, GPU e RAM para um jogo

6. A base de dados deve permitir inserir informação sobre quais componentes de hardware são suficientes para a execução de um jogo
7. A base de dados deve permitir a consulta de informação de todos os componentes anteriormente referidos, CPU, GPU e RAM
8. A base de dados deve permitir a consulta da informação dos utilizadores e as suas configurações de hardware criadas
9. A base de dados deve permitir principalmente consultar que componentes registados correm um determinado jogo
10. A base de dados deve permitir saber todos os jogos que correm numa determinada configuração
11. A base de dados de permitir saber dada uma configuração e um jogo, qual ou quais componentes não são suficientes para correr o jogo
12. A base de dados deve permitir por último saber se além de ser suficiente se dada configuração está dentro dos padrões recomendados para a execução de um jogo

2.2.2. Requisitos de exploração

Os seguintes requisitos identificam como é feito o povoamento e posterior utilização da informação inserida na base de dados

1. As configurações só são criadas caso existam previamente todos os componentes da mesma registados na base de dados
2. As configurações só podem ser criadas caso exista o utilizador ao qual se referem
3. Sempre que é inserido um novo jogo é criado um relacionamento entre esse jogo e todos os componentes de hardware que tem rating maior ou igual ao seu mínimo por meio de um *trigger*
4. Sempre que é inserido um novo componente de hardware é criado um relacionamento entre ele e todos os jogos que podem ser executados nele por meio de um *trigger*
5. A lista de componentes que correm um jogo é obtida através de um *procedure*

2.2.3. Requisitos de controlo

Os seguintes requisitos identificam como é manipulada a informação na base de dados e que acesso é concedido a cada tipo de utilizador

1. O administrador da BD pode inserir(INSERT), modificar(UPDATE), consultar(SELECT) e apagar (DELETE)

2. Os restantes utilizadores podem
    a. Inserir(INSERT), modificar(UPDATE) e consultar(SELECT) as suas próprias configurações
    b. Modificar(UPDATE) o seu nome
    c. Consultar(SELECT) os componentes de hardware
    d. Consultar(SELECT) a lista de jogos
    e. Realizar queries de consulta(SELECT) para saber se as suas configurações correm um jogo(s)
    f. Realizar queries de consulta(SELECT) relativas a jogos e quaisquer componentes de hardware registados na BD
    g. Inserir(INSERT) um novo utilizador na BD mas não apagar(DELETE)

2.3. Análise geral dos requisitos

Após análise geral deste levantamento de requisitos, é possível estipular a maneira correta de como a nossa base de dados se deve comportar. Aprofundamos também o conhecimento acerca do funcionamento da base de dados, o que nos permitiu criar os modelos e corrigir erros específicos que não cumpram ou complementam os requisitos.

Assim conseguimos assegurar que todas as relações entre os requisitos e as suas respetivas propriedades estão corretas, dando a resposta adequada ao que o administrador definiu como parâmetros para a implementação da nossa base de dados.

3. Modelação Conceptual

3.1. Apresentação da abordagem de modelação realizada

Foi optado por se fazer um diagrama ER de forma a nos ajudar, numa fase inicial, a criar o modelo conceptual. Um diagrama ER ilustra como e quais são as principais entidades, descritas por atributos e como se relacionam entre si sem, no entanto, haver qualquer consideração física para a criação desse modelo.

3.2. Identificação e caracterização de entidades

Tomou-se a decisão de se criar uma base dados que armazena toda a informação sobre as configurações dos utilizadores e sobre os componentes disponíveis no mercado ficando mais fácil a pesquisa e acesso de qualquer peça. A qualquer momento o gestor poderá adicionar nova informação à base de dados.

Um utilizador apresenta uma configuração ao "Will it run". Esta configuração composta por 3 componentes: a RAM, o GPU e o CPU, é o necessário para o site avaliar se o computador do utilizador consegue correr ou não um jogo da lista que o site disponibiliza. Qualquer dos componentes vai ser identificado pela marca, modelo, ano em que foi lançado e a velocidade. O CPU e o GPU terão ainda um rating associado que será depois exigido pelo jogo, o GPU tem ainda uma quantidade de VRAM e a RAM terá uma capacidade. O jogo terá identificado o nome e fabricante e ano em que foi lançado, e terá mínimos e recomendados exigidos para cada componente.

Com isto concluímos que iam ser preciso as entidades:

Entidade	Descrição	Ocorrência
Utilizador	Termo que descreve o indivíduo que testa a configuração no site.	Um Utilizador pode possuir várias configurações
Configuração	Termo que descreve as configurações de hardware que os Utilizadores criam e testam.	Uma configuração é possuída por um Utilizador e possui RAM, CPU e GPU
RAM	Termo que descreve a RAM do computador de uma configuração	A RAM faz parte de cada configuração e é suportada por

		jogos
CPU	Termo que descreve o CPU do computador de uma configuração	O CPU faz parte de cada configuração e é suportado por jogos
GPU	Termo que descreve o GPU do computador de uma configuração	O GPU faz parte de cada configuração e é suportado por jogos
Jogo	Termo que descreve os jogos que se podem testar no site	Os Jogos requerem RAM, GPU e CPU para funcionar.

Tabela 1 : Caracterização das Entidades

3.3. Identificação e caracterização dos relacionamentos

Depois de uma análise das entidades foram identificados os seguintes relacionamentos entre elas:

Utilizador e Configuração:

Um Utilizador submete uma ou várias configurações, e uma Configuração é submetida por um Utilizador. Este tipo de relacionamento entre Utilizador e Configuração é caracterizado por “possui”, e possui uma cardinalidade de 1 (Obrigatório) para N (Obrigatório).

Configuração e RAM:

Uma Configuração contém apenas um tipo de RAM e um tipo de RAM pode pertencer a várias configurações. Este tipo de relacionamento entre Configuração e RAM é caracterizado por “contém”, e possui uma cardinalidade de N (Obrigatório) para 1 (Obrigatório).

Configuração e CPU:

Uma Configuração contém apenas um CPU e um CPU pode pertencer a várias configurações. Este tipo de relacionamento entre Configuração e CPU é caracterizado por “contém”, e possui uma cardinalidade de N (Obrigatório) para 1 (Obrigatório).

Configuração e GPU:

Uma Configuração contém apenas um GPU e um GPU pode pertencer a várias configurações. Este tipo de relacionamento entre Configuração e GPU é caracterizado por “contém”, e possui uma cardinalidade de N (Obrigatório) para 1 (Obrigatório).

RAM e Jogo:

Um tipo de RAM pode ser compatível com vários jogos e um Jogo tem vários tipos de RAM com qual é compatível. Este tipo de relacionamento entre RAM e Jogo é caracterizado por “usado”, e possui uma cardinalidade de N (Obrigatório) para M (Obrigatório).

CPU e Jogo:

Um CPU é compatível com vários jogos e um Jogo tem vários CPU com qual é compatível. Este tipo de relacionamento entre CPU e Jogo é caracterizado por “usado”, e possui uma cardinalidade de N (Obrigatório) para M (Obrigatório).

GPU e Jogo:

Um GPU é compatível com vários jogos e um Jogo tem vários GPU com qual é compatível. Este tipo de relacionamento entre GPU e Jogo é caracterizado por “usado”, e possui uma cardinalidade de N (Obrigatório) para M (Obrigatório).

3.4. Identificação e caracterização das Associações dos Atributos com as Entidades e Relacionamentos

Entidade:Utilizador				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Número de identificação do Utilizador(PK)	Número inteiro positivo (INT)	Não	Identificador (Simples)
Nome	Nome do Utilizador	String variável(45)	Não	Simples

Tabela 2: Caracterização dos atributos da entidade Utilizador

Entidade:Configuração				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Número de identificação da Configuração (PK)	Número inteiro positivo (INT)	Não	Identificador (Simples)
Nome	Nome da Configuração	String variável(45)	Sim	Simples
Data	Data em que a configuração foi efectuada	Date	Não	Simples

Tabela 3: Caracterização dos atributos da entidade Configuração

Entidade: CPU				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Número de identificação do CPU(PK)	Número inteiro positivo (INT)	Não	Identificador (Simples)
Marca	Nome da marca do CPU	String variável(45)	Sim	Simples
Modelo	Nome do modelo do CPU	String variável(45)	Sim	Simples
Ano	Ano de lançamento do CPU	Número inteiro não negativo (INT)	Sim	Simples
Rating	Avaliação da performance do CPU	Número inteiro não negativo (INT)	Não	Simples
Velocidade	Velocidade de clock do CPU	Decimal(4,2)	Sim	Simples

Tabela 4: Caracterização dos atributos da entidade CPU

Entidade: GPU				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Número de identificação do GPU(PK)	Número inteiro não negativo (INT)	Não	Identificador (Simples)
Nome	Nome da marca do GPU	String variável até caracteres(45)	Sim	Simples
Marca	Nome da marca do GPU	String variável até caracteres(45)	Sim	Simples
Ano	Ano de lançamento do GPU	Número inteiro não negativo (INT)	Sim	Simples
Rating	Avaliação da performance do GPU	Número inteiro não negativo (INT)	Não	Simples
Velocidade	Velocidade de clock do GPU	Decimal(4,2)	Sim	Simples

Tabela 5: Caracterização dos atributos da entidade GPU

Entidade: RAM				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Nº identificação da RAM(PK)	Número inteiro não negativo (INT)	Não	Identificador (Simples)
Nome	Nome da RAM	String variável(45)	Sim	Simples
Marca	Nome da marca da RAM	String variável(45)	Sim	Simples
Ano	Ano de lançamento da RAM	Número inteiro não negativo (INT)	Sim	Simples
Capacidade	Número de GB da RAM	Número inteiro não negativo (INT)	Não	Simples
Velocidade	Velocidade de clock da RAM	Decimal(4,2)	Sim	Simples

Tabela 6: Caracterização dos atributos da entidade RAM

Entidade: Jogo				
Atributo	Descrição	Tipo de Dados e Domínio	Nulo	Tipo de Atributo
ID	Número de identificação do Jogo(PK)	Número inteiro não negativo (INT)	Não	Identificador (Simples)
Nome	Nome do jogo	String variável até caracteres(45)	Sim	Simples
Desenvolvedora	Nome do fabricante	String variável(45)	Sim	Simples
Ano	Ano de lançamento do Jogo	Número inteiro não negativo (INT)	Sim	Simples
CPUmin	Rating mínimo do CPU para executar o jogo	Número inteiro não negativo (INT)	Não	Simples
CPUrec	Rating recomendado do CPU para executar o jogo	Número inteiro não negativo (INT)	Não	Simples
GPUmin	Rating mínimo da GPU	Número inteiro não	Não	Simples

	para executar o jogo	negativo (INT)		
GPUrec	Rating mínimo da GPU para executar o jogo	Número inteiro não negativo (INT)	Não	Simples
RAMmin	Rating mínimo da RAM para executar o jogo	Número inteiro não negativo (INT)	Não	Simples
RAMrec	Rating recomendado da RAM para executar o jogo	Número inteiro não negativo (INT)	Não	Simples

Tabela 7:Caracterização dos atributos da entidade Jogo

3.5. Detalhe ou generalização de entidades

O modelo conceptual passou por várias transformações e reavaliações, numa primeira versão tínhamos os usuários diretamente ligados aos componentes de hardware num relacionamento 1-N, numa segunda avaliação já introduzimos a tabela configurações que agrupava todos os componentes numa configuração, que faz mais sentido

3.6. Apresentação e explicação do diagrama ER

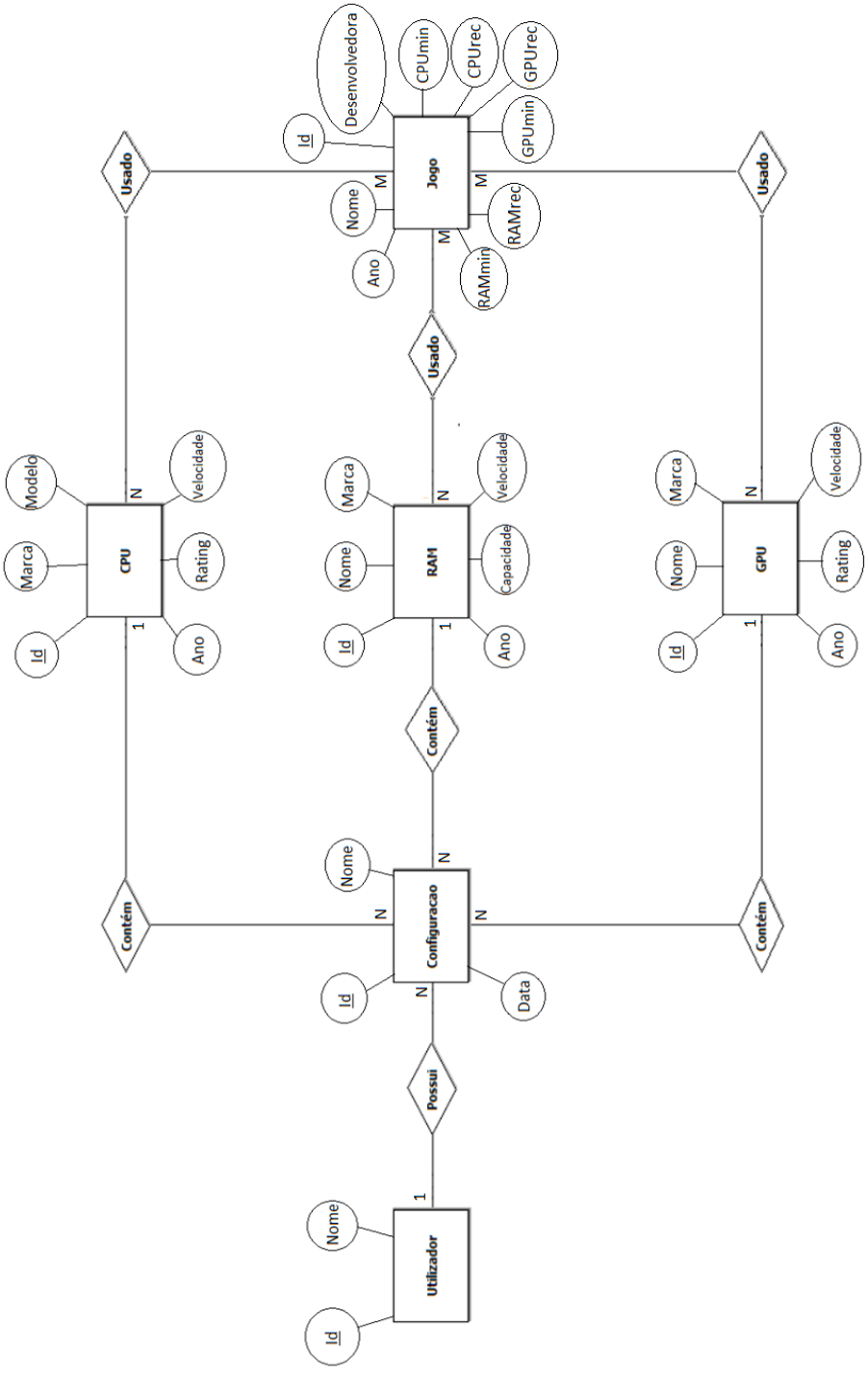

Figura 1 Diagrama ER

3.7. Validação do modelo de dados com o utilizador

A finalidade de uma base de dados é, no fim de contas, proporcionar maior facilidade e funcionalidade na gestão de informação à pessoa que a vai utilizar de modo a que durante o processo da criação da BD teremos sempre que garantir que os requisitos pedidos pelo utilizador são satisfeitos. O último passo de uma modelação conceptual consiste em validar o modelo por nós proposto junto do utilizador/gestor e fazer qualquer alteração sugerida. Usando o diagrama Entidade Relacionamento nessa revisão o modelo foi aprovado.

4. Modelação Lógica

4.1. Construção e validação do modelo de dados lógico

Uma vez feito o modelo conceptual procedemos para a criação do modelo lógico.

Como a nossa base de dados vai ter necessidade de guardar as configurações dos vários utilizadores começamos por criar uma tabela "User".

Esta tabela tem como atributo da chave primária o ID garantindo assim que cada User é único, podemos ainda guardar o nome do User como atributo simples. Posteriormente e como já mencionado há necessidade de guardar a configuração dos utilizadores do site. Vamos criar a tabela Configuração em que começamos por identificar o ID como chave primária de modo a garantir que não há diferentes entradas para a mesma configuração e dois atributos simples obrigatórios, o Nome da configuração e a Data.

A configuração vai ser relacionada com o User que a possui e com os componentes RAM, CPU E GPU, deste modo surgiu a necessidade de usar as chaves estrangeiras CPU_Id, RAM_Id, GPU_Id e User_Id que são herdadas das tabelas CPU, RAM, GPU e User respetivamente. Há necessidade de criar tabelas para cada um dos componentes necessários para testar um jogo, CPU, RAM e GPU. A tabela CPU tem como identificador único o Id(PK) e guardará ainda o Modelo, Marca, Ano, Rating e Velocidade como atributos simples. A tabela RAM também terá um Id único (PK) e atributos simples: Nome, Marca, Ano, Capacidade e Velocidade. A tabela GPU conta também com uma PK de Id e guarda ainda o Nome, Marca, Ano, Rating.

De seguida vamos precisar de guardar os jogos, como a relação entre os Jogos e a RAM, CPU e GPU é de N para M (um jogo tem vários CPUs que o podem correr e um CPU pode correr vários jogos) vai levar à criação de 3 tabelas intermediárias, **GPU_Jogo**, RAM Jogo e CPU Jogo.

A validação foi feita seguindo os parâmetros de integridade, nomeadamente a:

- integridade de entidade, que obtivemos ao exigir que todas as nossas chaves primárias fossem não nulas, utilizando o campo “NN” na criação do modelo com a ferramenta *MySQL WorkBench.*
- A integridade de chave, que diz que a chave primária é única, no caso selecionamos “PK” na ferramenta o que não permite entradas com a mesma chave.
- A integridade de domínio obtivemos quando indicamos na ferramenta o domínio para os atributos, por exemplo rating é do tipo “INT”

- Integridade referencial, isso foi feito introduzindo “FK” chaves estrangeiras onde as tabelas possuem uma “FK” por cada relacionamento com outra tabela, e essa “FK” corresponde as chaves primárias destas mesmas tabelas. Não temos nenhum caso de relacionamentos 1-1 , logo os 1-N tem a “PK” como “FK” na parte “N” do relacionamento, e no caso dos relacionamentos N-M fizemos tabelas intermediárias, como é o caso da tabela **“CPU_Jogo”** que contém as **“PK”** da tabela CPU e Jogo. O mesmo se aplica às outras duas tabelas intermédias.

Com isto tudo garantimos a integridade do modelo Lógico.

4.2. Desenho do modelo lógico

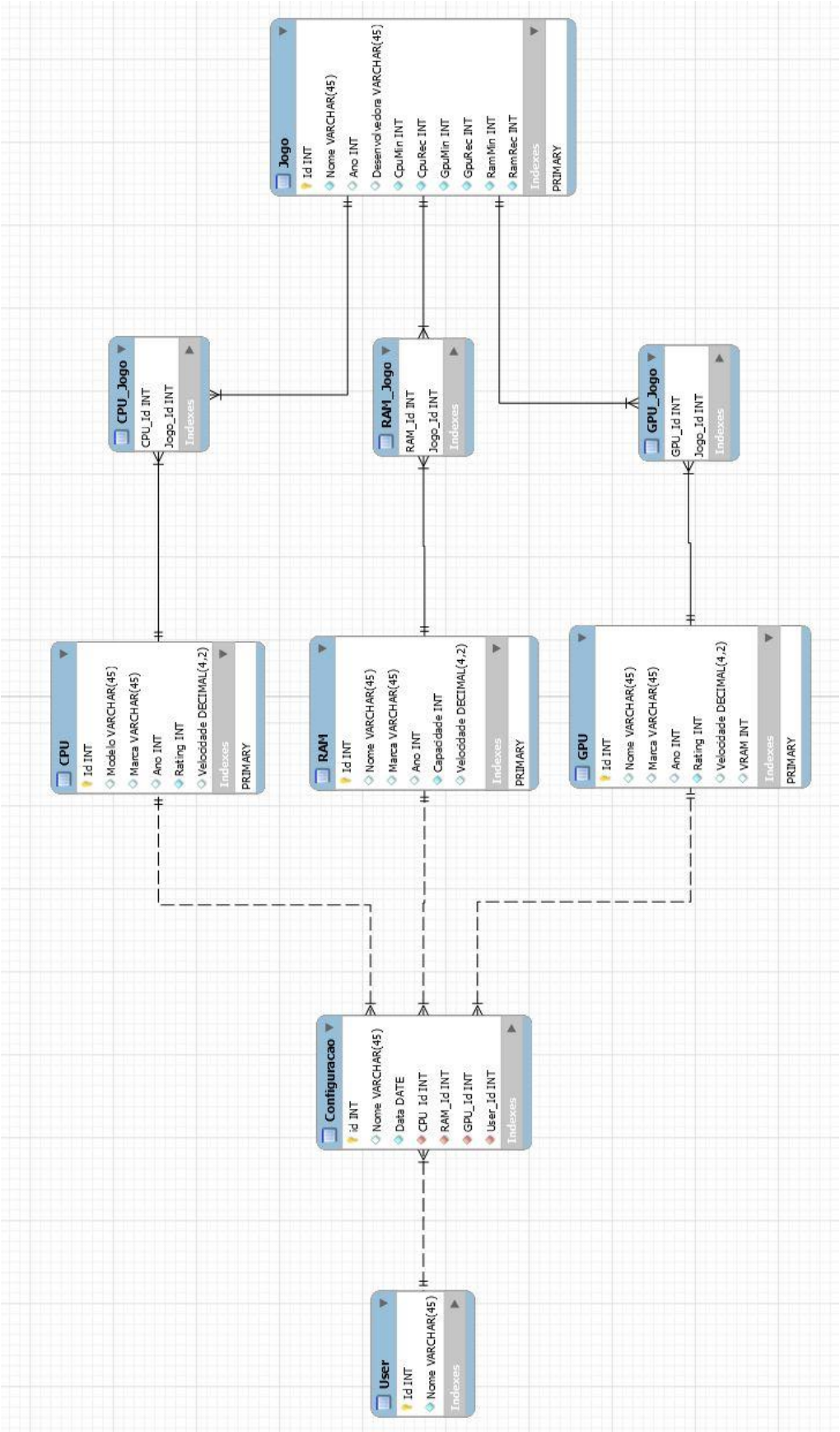

Figura 2: Modelo Lógico

4.3. Validação do modelo através da normalização

O objetivo da normalização é avaliar a qualidade do modelo minimizando a inconsistência e a redundância e tornando a BD mais percetível, simples e percetível. Vamos então conferir exemplos de regras de normalização que aplicámos:

1FN - 1ª Forma Normal

Para satisfazer a 1FN todos os atributos têm que ser atômicos, não pode haver informação repetida nem informação agrupada que pode ser separada.

No entanto existem alguns atributos que poderiam ser partidos, porém não foram porque foram simplificados já no modelo conceptual, ou achamos que não precisavam ser separados, nomeadamente:

Nome - O nome foi simplificado para ser atômico desde o início, logo não há entradas do tipo “Paulo Alves”, com nome e sobrenome.

Data - Realmente a data(de quando é criada uma configuração) poderia ser partida em ano, mês e dia porém como ao fazer o modelo lógico vimos que um tipo de dados é “Date”, que já é suportado pelo modelo lógico decidimos não o separar.

Modelo - O modelo, de todos os componentes, já tem no nome tanto o próprio modelo do componente como o nome da marca/fabricante , por exemplo “intel i7-7700k”, no entanto a decisão de primeiro, ter os atributos modelo e marca e segundo manter o nome da marca no modelo foram para poder procurar por marca/fabricante, caso não existisse o atributo marca nos componentes de hardware não o conseguimos fazer , mas também normalmente quando se refere a um componente é normal dizer o nome da marca primeiro, “olha tenho um intel pentium 4” ou “o meu computador tem uma nvidia gtx 1060 e uma corsair vengeance de 4GB”, logo decidimos manter no modelo o nome completo pelo qual é conhecido, acreditamos que essa pequena redundância não interfere na qualidade do modelo.

No entanto não permitimos a introdução de componentes sem marca no modelo, para evitar coisas como, modelo -> Intel i7-7700k e modelo ->I7-7700k, dois modelos iguais, porém introduzidos de forma diferente, é obrigatório que ao inserir seja respeitada essa regra de introduzir o modelo com a marca/fabricante no início.

2FN - 2ª Forma Normal

Uma tabela está na 1FN e todos os atributos não-chave são totalmente dependentes da chave primária.

No nosso modelo tas as tabelas tem como chave primária um id, não temos nenhuma chave primária que possa causar dependências parciais entre atributos e chave-primária.

3FN- 3ª Forma Normal

Uma tabela está na 2FN e nenhuma coluna não chave depende de outra coluna não chave.

O modelo satisfaz a 3FN, todos os atributos dependem do id de cada tabela e no caso das velocidades e capacidades nos componentes de hardware achamos que não dependem dos modelos visto que o mesmo modelo pode ter variações por overclocking ou undervolt de velocidade dos chips ou do consumo energético, ou por hardware do mesmo modelo mas sem exatamente as mesmas especificações como é o caso dos lançamentos de GPU onde são lançadas versões founders edition que tem diferentes velocidades.

4.4. Validação do modelo com interrogações do utilizador

Mostrar todos os GPUs que correm um jogo específico:

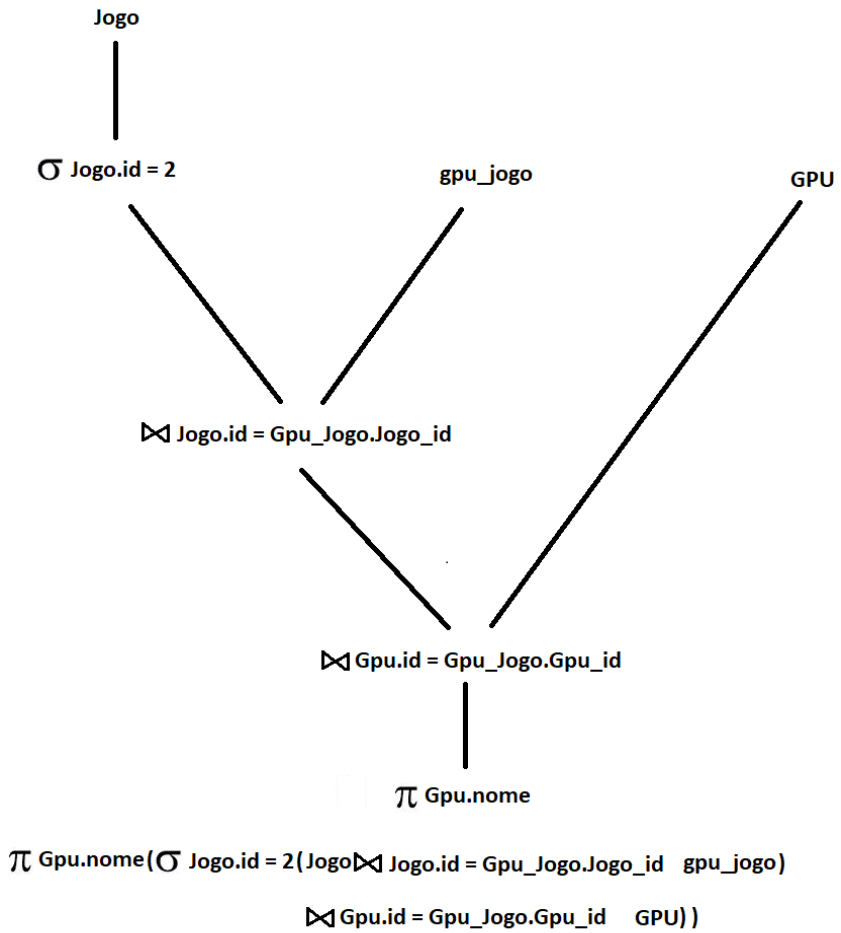

Figura 3 - Validação Modelo Interrogação

4.5. Revisão do modelo lógico com o utilizado

Mais uma vez no final do desenvolvimento do modelo lógico tivemos que rever o modelo com o utilizador de modo a que seja garantida a satisfação dos requisitos e das exigências do mesmo. Após a revisão do modelo com o utilizador este foi dado como aprovado e podemos seguir com o nosso trabalho.

5. Implementação Física

5.1. Seleção do sistema de gestão de bases de dados

Para a realização da nossa base de dados escolhemos utilizar o Sistema MySQL pelo facto de ser um dos melhores sistemas open-source disponíveis, não havendo necessidade de obrigar o empregador a despender de servidores e licenças pagas para outros sistemas closed-source.

Este sistema proporciona também alta disponibilidade, performance e consistência, sendo assim fácil e intuitivo de utilizar.

5.2. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------


-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`User`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `Id` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`CPU`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`CPU` (
  `Id` INT UNSIGNED NOT NULL,
  `Modelo` VARCHAR(45) NULL,
  `Marca` VARCHAR(45) NULL,
  `Ano` INT UNSIGNED NULL,
  `Rating` INT UNSIGNED NOT NULL,
  `Velocidade` DECIMAL(4,2) UNSIGNED NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;


```

```sql
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`RAM` (
  `Id` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Marca` VARCHAR(45) NULL,
  `Ano` INT UNSIGNED NULL,
  `Capacidade` INT UNSIGNED NOT NULL,
  `Velocidade` DECIMAL(4,2) UNSIGNED NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`GPU`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`GPU` (
  `Id` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Marca` VARCHAR(45) NULL,
  `Ano` INT UNSIGNED NULL,
  `Rating` INT UNSIGNED NOT NULL,
  `Velocidade` DECIMAL(4,2) UNSIGNED NULL,
  `VRAM` INT UNSIGNED NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Configuracao`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Configuracao` (
  `id` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NULL,
  `Data` DATE NOT NULL,
  `CPU_Id` INT UNSIGNED NOT NULL,
  `RAM_Id` INT UNSIGNED NOT NULL,
  `GPU_Id` INT UNSIGNED NOT NULL,
  `User_Id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_Configuracao_CPU1_idx` (`CPU_Id` ASC) VISIBLE,
  INDEX `fk_Configuracao_RAM1_idx` (`RAM_Id` ASC) VISIBLE,
  INDEX `fk_Configuracao_GPU1_idx` (`GPU_Id` ASC) VISIBLE,
  INDEX `fk_Configuracao_User1_idx` (`User_Id` ASC) VISIBLE,
  CONSTRAINT `fk_Configuracao_CPU1`
    FOREIGN KEY (`CPU_Id`)
    REFERENCES `mydb`.`CPU` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Configuracao_RAM1`
    FOREIGN KEY (`RAM_Id`)
    REFERENCES `mydb`.`RAM` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Configuracao_GPU1`
    FOREIGN KEY (`GPU_Id`)
    REFERENCES `mydb`.`GPU` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Configuracao_User1`
```

```
  CONSTRAINT `fk_Configuracao_User1`
    FOREIGN KEY (`User_Id`)
    REFERENCES `mydb`.`User` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Jogo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Jogo` (
  `Id` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Ano` INT UNSIGNED NULL,
  `Desenvolvedora` VARCHAR(45) NULL,
  `CpuMin` INT UNSIGNED NOT NULL,
  `CpuRec` INT UNSIGNED NOT NULL,
  `GpuMin` INT UNSIGNED NOT NULL,
  `GpuRec` INT UNSIGNED NOT NULL,
  `RamMin` INT UNSIGNED NOT NULL,
  `RamRec` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`CPU_Jogo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`CPU_Jogo` (
  `CPU_Id` INT UNSIGNED NOT NULL,
  `Jogo_Id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`CPU_Id`, `Jogo_Id`),
  INDEX `fk_CPU_has_Jogo_Jogo1_idx` (`Jogo_Id` ASC) VISIBLE,
  INDEX `fk_CPU_has_Jogo_CPU1_idx` (`CPU_Id` ASC) VISIBLE,
  CONSTRAINT `fk_CPU_has_Jogo_CPU1`
    FOREIGN KEY (`CPU_Id`)
    REFERENCES `mydb`.`CPU` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_CPU_has_Jogo_Jogo1`
    FOREIGN KEY (`Jogo_Id`)
    REFERENCES `mydb`.`Jogo` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `mydb`.`RAM_Jogo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`RAM_Jogo` (
  `RAM_Id` INT UNSIGNED NOT NULL,
  `Jogo_Id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`RAM_Id`, `Jogo_Id`),
  INDEX `fk_RAM_has_Jogo_Jogo1_idx` (`Jogo_Id` ASC) VISIBLE,
  INDEX `fk_RAM_has_Jogo_RAM1_idx` (`RAM_Id` ASC) VISIBLE,
  CONSTRAINT `fk_RAM_has_Jogo_RAM1`
    FOREIGN KEY (`RAM_Id`)
    REFERENCES `mydb`.`RAM` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_RAM_has_Jogo_Jogo1`
    FOREIGN KEY (`Jogo_Id`)
    REFERENCES `mydb`.`Jogo` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`GPU_Jogo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`GPU_Jogo` (
  `GPU_Id` INT UNSIGNED NOT NULL,
  `Jogo_Id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`GPU_Id`, `Jogo_Id`),
  INDEX `fk_GPU_has_Jogo_Jogo1_idx` (`Jogo_Id` ASC) VISIBLE,
  INDEX `fk_GPU_has_Jogo_GPU1_idx` (`GPU_Id` ASC) VISIBLE,
  CONSTRAINT `fk_GPU_has_Jogo_GPU1`
    FOREIGN KEY (`GPU_Id`)
    REFERENCES `mydb`.`GPU` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_GPU_has_Jogo_Jogo1`
    FOREIGN KEY (`Jogo_Id`)
    REFERENCES `mydb`.`Jogo` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5.3. Tradução das interrogações do utilizador para SQL

- Saber se uma configuração corre um certo jogo

```
CREATE FUNCTION ConsegueCorrer (configX INT, jogoX Int) -- DIZ SE CORRE E SE AO MIN OU REC
RETURNS VARCHAR(25)
BEGIN

        DECLARE res VARCHAR(20) ;
        set res = 'Nao corre';


        SELECT CASE

            WHEN (Jogo.CpuRec <= Cpu.Rating || Jogo.RamRec <= Ram.Capacidade || Jogo.GpuRec <= Gpu.Rating)  THEN 'Corre Recomendado'
            WHEN (Jogo.CpuRec > Cpu.Rating || Jogo.RamRec > Ram.Capacidade || Jogo.GpuRec > Gpu.Rating)     THEN 'Corre apenas Minimo'
            ELSE 0
        END

        INTO res FROM Configuracao

        INNER JOIN CPU ON Configuracao.CPU_id = Cpu.Id
        INNER JOIN GPU ON Configuracao.GPU_id = Gpu.Id
        INNER JOIN RAM ON Configuracao.RAM_id = Ram.Id

        INNER JOIN CPU_Jogo ON Cpu.Id = Cpu_Jogo.CPU_Id
        INNER JOIN GPU_Jogo ON GPU.Id = Gpu_Jogo.GPU_Id
        INNER JOIN RAM_Jogo on RAM.Id = Ram_Jogo.Ram_Id

        INNER JOIN Jogo ON CPU_Jogo.Jogo_Id = Jogo.Id && Ram_Jogo.Jogo_Id = Jogo.Id && Gpu_Jogo.Jogo_Id = Jogo.Id

        WHERE (Configuracao.Id = ConfigX && Jogo.Id=jogoX);

        RETURN res;

END$$
```

- Mostrar todos os GPU's que correm um certo jogo

```
CREATE PROCEDURE  GpuQueCorre (IN JogoX INT) -- MOSTRA TODOS OS GPUS QUE CORREM UM JOGO
BEGIN

        SELECT Gpu.Nome FROM Jogo
        INNER JOIN GPU_Jogo ON Gpu_Jogo.Jogo_Id = Jogo.Id
        INNER JOIN GPU ON Gpu_Jogo.GPU_id = Gpu.Id

        WHERE Jogo.Id = JogoX;

END$$
```

5.4. Escolha, definição e caracterização de índices em SQL

Em SQL, um índice define -se como uma estrutura de dados que permite ao sistema de gestão de base de dados localizar registos num ficheiro de uma maneira eficiente e rápida, melhorando o tempo de resposta.

Os índices utilizados na nossa base de dados são apenas as chaves primárias e estrangeiras associadas às tabelas criadas. Visto ser uma BD de pequena escala, não vimos necessidade em definir índices extras e desta maneira acabamos por melhorar o tempo de resposta das queries.

5.5. Estimativa do espaço em disco da base de dados e taxa de crescimento anual

Com o sistema MySQL, o motor de armazenamento utilizado é o InnoDB.

Após uma pesquisa ficamos a saber o espaço que ocupa cada tipo de domínio:

- Integer (4 bytes)
- Tinyint (1 byte)
- Mediumint (3 bytes)
- Decimal (4 bytes)
- Date (3 bytes)
- Datetime (5 bytes)
- Time (1 byte)
- Varchar(N) (1+N bytes)
- Text ((2^16 + 2) bytes)

Partindo da seguinte query ficamos a saber o espaço que cada tabela da nossa BD ocupa (em Kb):

```
SELECT TABLE_NAME AS "Table Name",
table_rows AS "Quant of Rows", ROUND( (data_length + index_length) /1024, 2 ) AS "Total Size Kb"
FROM information_schema.TABLES
WHERE information_schema.TABLES.table_schema = 'mydb'
LIMIT 0 , 30
```

Table Name	Quant of Rows	Total Size Kb
configuracao	2	80.00
cpu	6	16.00
cpu_jogo	22	48.00
cpuquecorrev	0	0.00
gpu	5	16.00
gpu_jogo	18	48.00
jogo	4	16.00
quaiscorrev	0	0.00
ram	3	16.00
ram_jogo	10	48.00
user	5	16.00

Figura 4 - Tabela de Espaço Ocupado

Numa estimativa grosseira estamos a utilizar 70Kb por entrada.

No entanto como os jogos que costumam precisar de hardware mais “potente” são jogos de maior dimensão, os considerados “AAA” e estes têm lançamentos na ordem das dezenas por ano o crescimento anual de entradas de jogos não seria muito grande, caso fossemos inserir todo tipo de jogos incluindo os de menor dimensão a história já seria diferente.

A quantidade de novos modelos de RAM, CPU e GPU também não passa da casa das dezenas por ano, porém em bastante menor quantidade do que jogos, neste momento há poucas empresas que fazem lançamento de novos CPU e GPU, o mercado é quase monopolizado por meia dúzia delas, e o resto não é importante para o nosso sistema de verificação de jogos que correm em hardware mainstream

Analisando estes dados podemos concluir que o espaço ocupado não vai crescer mais que uns poucos MB(MegaByte) por ano, o que não representa nenhum problema atualmente onde a conversa está sempre na ordem de GB(GigaByte) pra cima.

5.6. Definição e caracterização das vistas de utilização em SQL

Para na nossa base de dados, tomamos a decisão de criar duas views que proporcionam informação pertinente ao utilizador. Apresentamos de seguida o código SQL das views criadas:

- Tabela que apresenta todos os Jogos que podem correr numa determinada Configuração:

```
CREATE VIEW  QuaisCorreV AS
SELECT Jogo.Nome, Configuracao.Id FROM Configuracao


        INNER JOIN CPU ON Configuracao.CPU_id = Cpu.Id
        INNER JOIN GPU ON Configuracao.GPU_id = Gpu.Id
        INNER JOIN RAM ON Configuracao.RAM_id = Ram.Id

        INNER JOIN CPU_Jogo ON Cpu.Id = Cpu_Jogo.CPU_Id
        INNER JOIN GPU_Jogo ON GPU.Id = Gpu_Jogo.GPU_Id
        INNER JOIN RAM_Jogo on RAM.Id = Ram_Jogo.Ram_Id

        INNER JOIN Jogo ON CPU_Jogo.Jogo_Id = Jogo.Id && Ram_Jogo.Jogo_Id = Jogo.Id && Gpu_Jogo.Jogo_Id = Jogo.Id $$


-- Select * From QuaisCorreV  Where Id = 1;
```

- **Tabela que apresenta todos os CPUs que correm um determinado Jogo**

```
CREATE view  CpuQueCorreV AS


        SELECT Cpu.Modelo, Jogo.Id FROM Jogo
        INNER JOIN CPU_Jogo ON Cpu_Jogo.Jogo_Id = Jogo.Id
        INNER JOIN CPU ON Cpu_Jogo.CPU_id = Cpu.Id
        $$
```

5.7. Definição e caracterização dos mecanismos de segurança em SQL

```
CREATE USER 'BdAdmin'@'localhost';
CREATE USER 'BdUser'@'localhost';



GRANT ALL PRIVILEGES ON mydb TO 'BdAdmin'@'localhost';

GRANT INSERT, UPDATE ON Configuracao to 'BdUser'@'localhost';
GRANT INSERT, UPDATE ON user to 'BdUser'@'localhost';
```

O admin pode alterar todas as tabelas, e um user normal só pode alterar e inserir na sua tabela de configurações ou na sua própria, nomeadamente modificar o seu nome, além disso não pode fazer qualquer modificação na base de dados, só consultar informação

5.8. Revisão do sistema implementado com o utilizador

Esta última fase do nosso projeto foi focada na modelação física da BD, neste caso levou-se em conta as limitações impostas pelo SGBD escolhido e foi criado com base nos exemplos de modelagem de dados produzidos no modelo lógico. Mais uma vez, foi feita uma revisão do sistema com o utilizador de modo a ser aceite, garantindo o cumprimento de todos os requisitos inicialmente previstos.

6. Conclusões e Trabalho Futuro

Concluído o projeto da conceção e validação da base de dados do “Will it run?” resta-nos a criação do website que permita a exploração da informação guardada.

O website teria que possibilitar com registo prévio de um utilizador a que este possa inserir e manipular informação sobre as suas configurações e principalmente comparar os componentes que tem com os que são necessários para correr determinado jogo.

O mesmo website também podia permitir ao utilizador guardar uma lista dos jogos que possui e de um histórico de configurações que já teve e das modificações que fez.

O mesmo website deveria permitir que mesmo sem registo e login que um utilizador escolha uma configuração no momento, utilizado os componentes registados e pesquisar quais jogos seriam compatíveis, neste caso poderíamos optar por guardar num cookie do browser essa informação.

Outra implementação possível seria a de dar possibilidade a um utilizador escolher um jogo, ver os seus requisitos mínimos e escolher um hardware que satisfaz esse requisito e ser redirecionado a uma loja ou lojas que o vendem.

No entanto para o âmbito da UC de base de dados damos por concluído com sucesso todo o processo de forma satisfatória.

7. Referências Bibliográficas

1- Database Systems - A Practical Approach to Design, Implementation, and Management SIXTH EDITION by Thomas Connolly and Carolyn Begg (2014)