



Universidade do Minho
Licenciatura em Engenharia Informática

Unidade Curricular de Bases de Dados

Ano Letivo de 2022/2023

Loja de videojogos do Sr. Clamídeo

Carlos Eduardo Culolo Dantas da Costa, A88551

Simão Paulo da Gama Castel-Branco e Brito, A89482

Tomás Cardoso Francisco, A93193

junho, 2023

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Loja de videojogos do Sr. Clamídeo

Carlos Eduardo Culolo Dantas da Costa, A88551

Simão Paulo da Gama Castel-Branco e Brito, A89482

Tomás Cardoso Francisco, A93193

junho, 2023

Resumo

Este projeto tem como propósito o desenvolvimento de um sistema de gestão de uma loja de videojogos em segunda mão. O principal objetivo é criar uma base de dados robusta e eficiente, capaz de armazenar informações sobre clientes, colaboradores, jogos e transações e implementar o sistema capaz de a gerir através de funcionalidades, como o registo de transações, clientes ou jogos ou operações de consulta.

A implementação foi dividida em várias etapas. Inicialmente é definido o sistema, através da contextualização e fundamentação do mesmo, indicação das principais motivações, definição de objetivos, análise da viabilidade do desenvolvimento do projeto e estruturação da equipa de trabalho, recursos necessários e plano de execução. Segue-se o levantamento e a análise dos requisitos do sistema, fundamentais para a modelação conceptual do mesmo, na qual identificámos entidades, relacionamentos e atributos relevantes. Com a tradução do diagrama ER para um modelo lógico, identificamos todas as relações necessárias para a base de dados, bem como os relacionamentos entre elas.

Com a implementação física, colocamos em prática todos os procedimentos e modelos teóricos produzidos, criando tabelas, queries capazes de responder às principais interrogações dos utilizadores do sistema, vistas de utilização capazes de simplificar o acesso aos dados e melhorar a eficiência das operações de consulta, procedimentos capazes de povoar a base de dados e perfis de utilizador que garantem a segurança. Criamos ainda um sistema de recuperação de dados, que funciona através da criação de scripts de backup. Por fim, expomos ainda a implementação de um Sistema de Recolha de Dados capaz de povoar a nossa Base de Dados.

Ao longo deste relatório, mostramos a utilização de ferramentas como o MySQL e o brModelo e de linguagens como SQL e Python.

Área de Aplicação: Desenho e arquitetura de Sistemas de Bases de Dados; Script Python

Palavras-Chave: Gestão de Projetos; Bases de Dados Relacionais; Definição de Sistemas; Desenvolvimento de Sistemas; Requisitos de Sistemas; Modelação de Dados; Modelo Conceptual; Modelo Lógico; brModelo; Arquitetura de Bases de Dados; Integridade de Dados; Restrições e Regras de Validação; Queries; Manipulação de Dados; SQL; MySQL; Python; SGBD; Backup; Recuperação de Dados; Indexação; Segurança de Dados.

Índice

Resumo	i
Índice	ii
Índice de Figuras	iv
Índice de Tabelas	vi
1. Definição do sistema	1
1.1. Contexto de aplicação e Fundamentação do sistema	1
1.2. Motivação e Objetivos	2
1.3. Análise da viabilidade do processo	2
1.4. Recursos e Equipa de Trabalho	3
1.4.1 Recursos	3
1.4.2 Equipa de Trabalho	4
1.5. Plano de Execução do projeto	4
2. Levantamento e Análise de Requisitos	6
2.1. Método de levantamento e de análise de requisitos adotado	6
2.2. Organização dos requisitos levantados	7
2.3. Análise e validação geral dos requisitos	10
3. Modelação Conceptual	11
3.1. Apresentação da abordagem de modelação realizada	11
3.2. Identificação e caracterização das entidades	12
3.3. Identificação e caracterização dos relacionamentos	13
3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos	14
3.5. Apresentação e explicação do diagrama ER produzido	16
4. Modelação Lógica	17
4.1. Construção e validação do modelo de dados lógico	17
4.2. Normalização de Dados	19
4.3. Apresentação e explicação do modelo lógico produzido	20
4.4. Validação do modelo com interrogações do utilizador	21
5. Implementação Física	23

5.1. Tradução do esquema lógico para o sistema de gestão de bases de dados escolhido em SQL	23
5.2. Tradução das interrogações do utilizador para SQL	27
5.3. Definição e caracterização das vistas de utilização em SQL	29
5.4. Cálculo do espaço da Base de Dados	30
5.5. Indexação do Sistema de Dados	32
5.6. Procedimentos implementados	33
5.7. Plano de segurança e recuperação de dados	36
6. Implementação do Sistema de Recolha de Dados	39
6.1. Apresentação e modelo do sistema	39
6.2. Implementação do sistema de recolha	40
6.3. Funcionamento do sistema	42
7. Conclusões e Trabalho Futuro	44

Índice de Figuras

Figura 1 - GANTT (Fases 1,2 e 3)	5
Figura 2 - GANTT (Fases 4-7)	5
Figura 3 - Diagrama ER	16
Figura 4 - Modelo Lógico	20
Figura 5 - Criação da tabela Jogo	24
Figura 6- Criação da tabela Cliente	24
Figura 7 - Criação da tabela Colaborador	24
Figura 8 - Criação da tabela Transação	25
Figura 9 - Criação da tabela TransaçãoContem	25
Figura 10 - Criação das tabelas referentes às relações criadas para atributos multi-valor	26
Figura 11 - Listagem de jogos em stock	27
Figura 12 - Consulta da quantidade e valor total transacionados para um dado jogo em uma determinada transação	27
Figura 13 - Função para obter média dos preços dos jogos para ano e plataforma dados	27
Figura 14 - Procedimento para obter a lista de jogos transacionados por um dado cliente	28
Figura 15 - Procedimento para obter os nomes dos clientes que compraram jogos a partir de um certo preço	28
Figura 16 - Procedimento que lista transações de um cliente num intervalo de tempo, ordenadas decrescentemente por data de transação	28
Figura 17 - Procedimento que lista as 10 transações de maiores valores	29
Figura 18 - Procedimento que lista os 10 jogos mais vendidos de sempre da loja	29
Figura 19 - Vista de resumo de transações por cliente	29
Figura 20 - Vista dos jogos mais populares	30
Figura 21 - Vista de Disponibilidade de Stock	30
Figura 22 - Código SQL para a criação dos índices	32
Figura 23 - Procedimento para registar um novo cliente	33

Figura 24 - Procedimento para registar um novo jogo	34
Figura 25 - Procedimento para registar uma Transação	35
Figura 26 - Código SQL para a criação do perfil Clamídeo	36
Figura 27 - Código SQL para a criação do perfil Funcionário	36
Figura 28 - Código SQL para a criação do perfil Utilizador	36
Figura 29 - Conteúdo do ficheiro backupScript.bat	37
Figura 30 - Execução do backupScript.bat	37
Figura 31 - Excerto inicial do ficheiro de backup .sql	38
Figura 32 - Povoamento das tabelas Cliente, Jogo, ClienteEmail, ClienteContacto, Jogo_Géneros e Jogo_Desenvolvedores	40
Figura 33 - Povoamento da tabela Colaborador	40
Figura 34 - Povoamento das tabelas Transação e TransaçãoContem	40
Figura 35 - Estabelecimento da conexão	41
Figura 36 - código python para povoamento da tabela Jogo	41
Figura 37 - Conteúdo inserido na tabela <i>Jogo</i>	42
Figura 38 - Últimos registos da tabela <i>Jogo_Desenvolvedores</i>	42
Figura 39 - Conteúdo das tabelas Transação e TransaçãoContem	43

Índice de Tabelas

Tabela 1 - Recursos necessários para o desenvolvimento do SBD	3
Tabela 2 - Entidades	13
Tabela 3 – Relacionamentos	13
Tabela 4 – Atributos	15

1. Definição do sistema

A loja de videojogos em 2ª mão do Sr. Clamídeo é um estabelecimento no qual, para além de se encontrarem os melhores preços de videojogos do retalho tradicional, se encontram também as ofertas mais apelativas para todos aqueles que se queiram ver livres de jogos que tenham arrumados na estante e para os quais já não lhes deem nenhum uso. O Sr. Clamídeo precisa urgentemente de implementar na sua loja uma Base de Dados que lhe permita melhorar imenso a organização e relacionamento entre as informações relativas aos clientes, aos artigos que tem na loja e às transações efetuadas, sejam estas de compra ou venda de jogos.

O desenvolvimento do Sistema de Bases de Dados que vai suportar não só a Base de Dados em si, mas também o Sistema de Gestão de Bases de Dados que vai ajudar os Administradores do Sistema a controlarem a própria Base de Dados, foi dividido em 7 etapas: Definição do Sistema; Levantamento e Análise de Requisitos; Modelação Conceptual; Modelação Lógica; Implementação Física; Implementação do Sistema de Recolha de Dados; Implementação do Sistema de Painéis de Análise.

Nesta primeira fase procedemos à definição do sistema dentro daquilo que é o contexto do nosso caso de estudo – a loja de videojogos – e os objetivos pretendidos pelo Sr. Clamídeo com a implementação deste sistema. No fim deste tópico procedemos ainda ao delineamento do trabalho a realizar, distribuindo tarefas e definindo metas e prazos a cumprir.

1.1. Contexto de aplicação e Fundamentação do sistema

Situada no coração de Braga, a loja de Alberto Clamídeo é um ícone local com mais de 30 anos de serviço ininterrupto à cidade Augusta. Reconhecida pelos preços baixos praticados e pela qualidade e raridade dos produtos oferecidos, a loja é também muito bem reputada devido à amabilidade do Sr. Clamídeo com todos que entram no seu acolhedor estabelecimento na Rua do Souto.

Inicialmente, esta era uma loja de venda e troca de cassetes, mas, graças à mente visionária do Sr. Clamídeo, com a viragem do século a loja passou a ter nos videojogos o seu principal produto. No entanto, nos últimos anos verificou-se um aumento considerável na aquisição de jogos em formato digital diretamente nas consolas, o que fez com que a compra de jogos físicos se tenha tornado numa atividade direcionada sobretudo para colecionadores e saudosistas. Motivado pela necessidade de inovar, Alberto Clamídeo decidiu fazer a revisão total do seu stock.

1.2. Motivação e Objetivos

Após a revisão, o Sr. Alberto concluiu que ainda possuía em stock bastantes jogos antigos pouco procurados pelos clientes e também jogos que não se recordava sequer de ter vendido. Além disso, ele não tinha conhecimento da quantidade exata de jogos que possuía em stock nem da quantidade de clientes ou das suas frequências de transações na loja.

Coincidentemente, o seu primogénito Robertinho Clamídeo frequentava a licenciatura em Engenharia Informática na Universidade do Minho, pelo que o Sr. Alberto lhe pediu uma sugestão que pudesse ajudar a resolver estes aspetos negativos do método de funcionamento da loja. Matriculado na Unidade Curricular de Bases de Dados, o Robertinho rapidamente se apercebeu de que o que poderia ajudar a inverter a trajetória descendente da loja seria a implementação de um Sistema de Bases de Dados, capaz de ajudar na gestão e organização do inventário e das informações dos clientes e que iria permitir uma melhor análise dos dados relativos às compras e vendas de artigos. Entusiasmado com a perspectiva de trabalhar num projeto concreto onde pudesse aplicar os seus conhecimentos dessa disciplina e com a aprovação do pai, o Robertinho prontamente recrutou dois colegas, formando assim a equipa de desenvolvimento deste sistema, o qual o Sr. Clamídeo tem confiança que possa modernizar a loja de modo que esta satisfaça a procura dos clientes já existentes e possa atrair novos fregueses.

Com a implementação do Sistema de Bases de Dados, o Sr. Clamídeo pretende:

- Possuir uma lista fidedigna de todo o seu stock;
- Gerir de forma eficiente os seus lucros e faturasções;
- Contabilizar todas as entradas e saídas de produtos da loja;
- Ter um registo de todos os clientes que já efetuaram alguma compra e/ou venda de jogos na loja;
- Poder traçar tendências nas transações dos seus produtos e, consequentemente, refinar o seu modelo de negócio.

Após a implementação do sistema, o Sr. Clamídeo pretende começar a comercializar os jogos em segunda mão através da internet, por encomenda. Para tal, é-lhe extremamente necessário que tenha o seu stock contabilizado.

1.3. Análise da viabilidade do processo

O Sr. Clamídeo pretende que o sistema seja simples e de fácil utilização e está convencido que a implementação de um sistema de bases de dados pode trazer os seguintes benefícios para a sua loja:

- Melhorar a gestão do inventário tornando-a mais eficiente e permitindo que ele conheça com precisão o número de artigos em stock e que jogos precisam ser reabastecidos. Isto **evitará a perda de vendas** por falta de stock ou excesso de stock de jogos que não vendem bem, o que contribuirá para **aumentar os lucros**.
- Permitir acompanhar o histórico de vendas de jogos, incluindo quais os jogos que vendem melhor e em que alturas eles são mais procurados e quais os jogos que são menos populares. Isto ajudará o Sr. Clamídeo a **tomar decisões informadas** sobre quais jogos comprar no futuro, o que permitirá evitar gastos desaproprados em jogos que não vendem tanto.

- Tomar **decisões de negócios informadas** e **maximizar a eficiência** da loja, através da criação de relatórios e análises que ajudem a identificar tendências e oportunidades de venda.
- Ajudar a loja a gerir o relacionamento com os clientes, permitindo o armazenamento de informações sobre os mesmos (nome, contacto, email, histórico de compras, etc.) o que iria **melhorar a forma como a loja comunica com os clientes**. Iria também permitir ao Sr. Clamídeo **acompanhar as preferências dos clientes** e, então, **melhorar a satisfação dos clientes, personalizando a experiência** dos mesmos ao recomendar jogos que eles possam gostar com base no seu histórico de compras.

Foi ainda requerida pelo Sr. Clamídeo ao grupo que vai desenvolver o sistema a realização de uma análise detalhada dos custos envolvidos na implementação do sistema e dos custos contínuos de manutenção e atualização do mesmo que lhe permitisse determinar se os benefícios superariam os custos. Dado que um dos elementos do grupo já obteve frequência a uma Unidade Curricular na área da Economia, os seus conhecimentos nessa área permitiram-lhe realizar esta análise, verificando que, de facto, os benefícios ultrapassarão imenso os custos no médio-longo prazo.

Por fim, foi também estabelecido, e assinado por todas as partes intervenientes no desenvolvimento do sistema, o "Acordo de Confidencialidade e Privacidade" do projeto que garante a segurança dos dados dos clientes e a conformidade com as leis de privacidade dos dados. Este Acordo foi elaborado e certificado pela esposa do Sr. Clamídeo, a Dona Elisete, licenciada em Direito e notária de profissão.

1.4. Recursos e Equipa de Trabalho

1.4.1 Recursos

Os recursos necessário para o desenvolvimento deste Sistema podem ser divididos da seguinte forma:

RECURSOS HUMANOS	RECURSOS MATERIAIS
<ul style="list-style-type: none"> • <u>Equipa de desenvolvimento</u>, composta por 4 alunos da licenciatura em Engenharia Informática na Universidade do Minho; • Sr. Clamídeo, <u>proprietário e responsável</u> da loja; • Serginho, <u>colaborador</u> da loja; • <u>Clientes</u>. 	<ul style="list-style-type: none"> • <u>Hardware</u>: 1 <u>servidor</u> para hospedar o sistema, 2 <u>computadores</u> de loja, <u>dispositivos de armazenamento</u> (discos rígidos externos e/ou internos, pendrives, etc.), <u>equipamento de rede</u> (1 router), 1 impressora (permitirá imprimir relatórios do sistema) e <u>1 tablet/smartphone</u> (permitirá aceder ao sistema a partir de um dispositivo móvel). • <u>Software</u>: Sistema operativo, Sistema de Gestão de Bases de Dados (SGBD), Sistema de gestão de Inventário e Vendas, software de segurança (firewall, antivírus, etc.) e ferramentas de backup e recuperação dos dados.

Tabela 1 - Recursos necessários para o desenvolvimento do SBD

1.4.2 Equipa de Trabalho

A equipa de trabalho foi dividida em 3 “subequipas”:

1.1 Pessoal Interno:

- Sr. Clamídeo – Funcionamento da loja, registo das compras e vendas de jogos, atualização de inventário, gestão do stock existente (stock à venda em loja e stock em armazém).
- Colaborador (Serginho) – Registo das compras e vendas de jogos e atualização de inventário.

2.1 Pessoal externo:

- Equipa de desenvolvimento (alunos de LEI) – Levantamento de requisitos, modelação do sistema, implementação do sistema e manutenção (futura) do sistema.

3.1 Outros:

- Clientes – Feedback sobre a qualidade da loja quanto à sua organização e aos jogos que são lá vendidos.

1.5. Plano de Execução do projeto

A implementação de um sistema de bases de dados é um projeto que exige uma cuidadosa planificação e gestão. Para garantir o seu sucesso, é fundamental ter um plano de execução bem definido que estabeleça claramente as tarefas a realizar, os prazos estabelecidos e os seus intervenientes.

Nesta secção, apresentamos então o nosso plano de execução, mapeado num diagrama de GANTT que permitirá visualizar todas as tarefas – agrupadas pelas sete fases nas quais o desenvolvimento do projeto foi dividido – bem como os responsáveis por cada tarefa, o estado do progresso atual de cada uma delas (que indica se as tarefas já estão concluídas, se estão em execução ou se ainda estão por iniciar), a data de início da tarefa e a sua duração. Com estes dois últimos indicadores, foram construídas automaticamente as barras visuais apresentadas no gráfico.

Implementação do Sistema de Bases de Dados

Loja de Jogos em Segunda Mão do Sr. Clamídeo

Sr. Clamídeo

Início do projeto:

20/02/2023

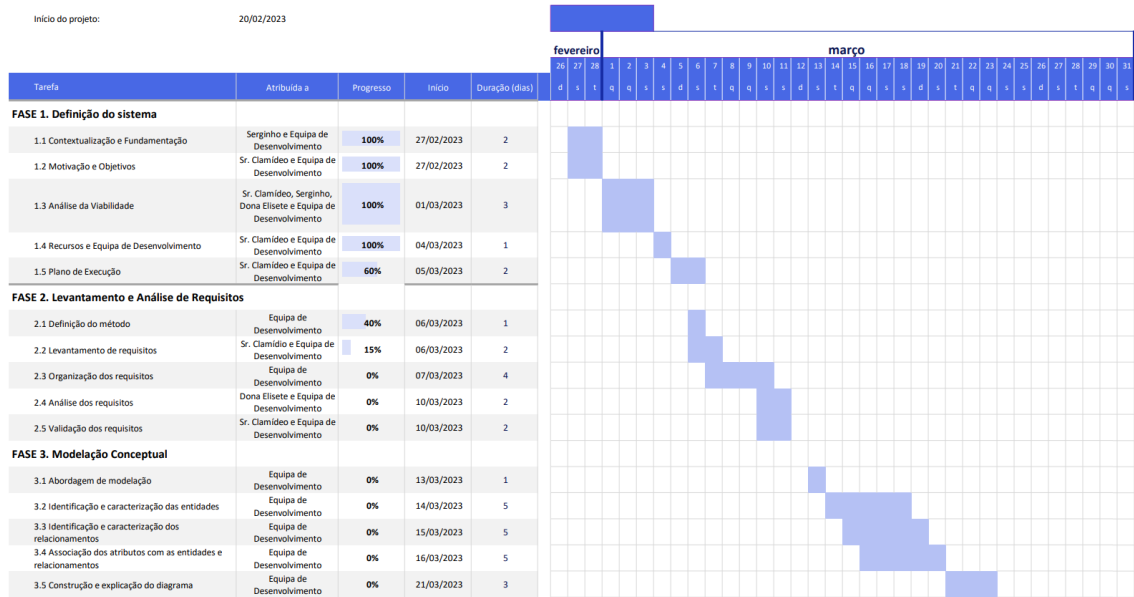


Figura 1 - GANTT (Fases 1,2 e 3)

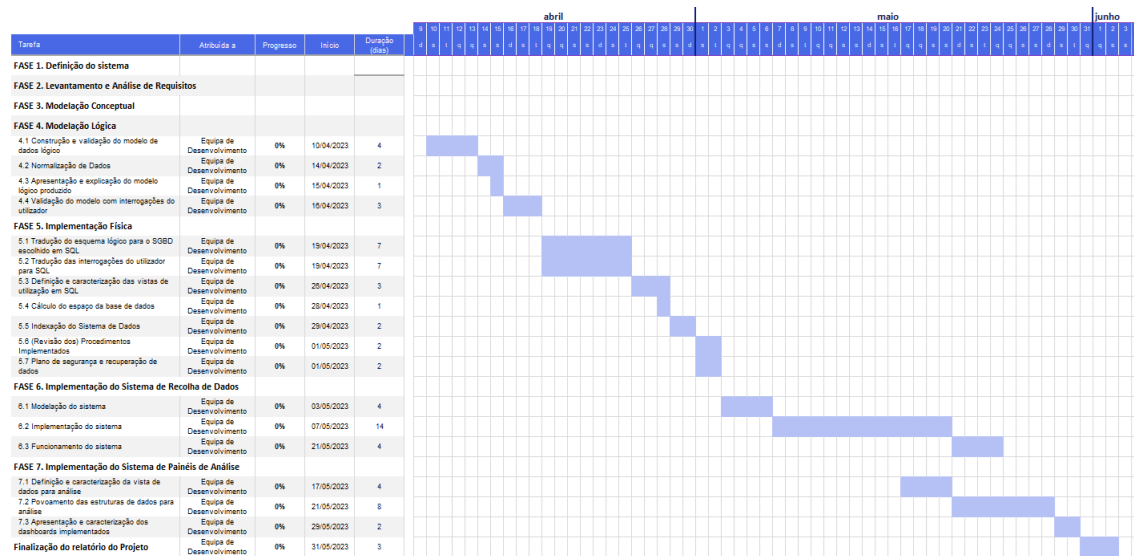


Figura 2 - GANTT (Fases 4-7)

2. Levantamento e Análise de Requisitos

Tendo o caso de estudo devidamente apresentado e delineado, procedemos então ao levantamento de requisitos que descrevem os processos existentes no contexto da loja do Sr. Clamídeo. O objetivo desta fase é entender o ambiente em que o sistema será utilizado, identificando as necessidades do(s) utilizador(es) e estabelecendo os requisitos que o sistema deve cumprir para atender a essas mesmas necessidades. Esta etapa é crucial para o desenvolvimento do projeto, pois é nela onde criamos a base para o desenrolar do mesmo e garantimos que o produto final atenderá às expectativas do Sr. Clamídeo. O sucesso do sistema depende em grande parte da qualidade dos requisitos levantados e da sua correta análise e validação.

2.1. Método de levantamento e de análise de requisitos adotado

Primeiramente, é necessário estabelecer o(s) método(s) que vamos utilizar para realizar o levantamento, a análise e a organização dos requisitos.

Iremos então utilizar os seguintes procedimentos no processo de identificação dos requisitos:

- **Diálogo** com o Sr. Clamídeo e com o Serginho onde serão descritos os processos relativos às transações efetuadas na loja, ajudando-nos a entender as necessidades e expectativas do proprietário e do colaborador da loja, que são os principais utilizadores do sistema que pretendemos desenvolver.
- **Observação de campo**, ou seja, a observação durante algumas horas de todas as ocorrências dentro da loja, o que permitirá ter uma noção daquilo que é o dia-a-dia neste contexto. Este método permitir-nos-á compreender os processos existentes e identificar problemas e oportunidades de melhoria.
- **Análise de documentação e sistema existentes**, que sabemos à partida serem muito tradicionais e rudimentares (dossier físico dividido em 2 partes: stock existente e transações). Esta inspeção permitirá entender o funcionamento da loja e que informações são importantes para o Sr. Clamídeo no contexto do stock existente e das transações de artigos.

- **Questionários** realizados aos clientes de modo a obter a opinião – anónima e sincera – destes no que diz respeito à loja do Sr. Clamídeo. Procuramos também saber a razão da preferência dos clientes por esta loja. A opinião dos fregueses é importante para definir requisitos que atendam às suas necessidades e, assim, melhorar a experiência geral de compra e/ou venda na loja.

Estando os requisitos identificados, é necessário então proceder a uma análise e organização dos mesmos. A revisão tem como principal objetivo identificar e eliminar eventuais erros, inconsistências, redundâncias e ambiguidades. Em relação à organização dos requisitos, estes serão divididos em três tipos: **Requisitos de Descrição**, **Requisitos de Exploração** e **Requisitos de Controlo**. Decidimos também atribuir prioridades aos requisitos, classificando-os como “críticos” ou apenas “desejáveis” atendendo aos seguintes critérios:

- Requisitos que são obrigatórios para o funcionamento básico do sistema são priorizados em relação aos opcionais.
- Requisitos que atendem às necessidades mais críticas do Sr. Clamídeo e dos clientes da loja foram priorizados em relação a outros requisitos menos críticos.
- Requisitos que podem ser facilmente implementados foram priorizados em relação aos mais complexos.

2.2. Organização dos requisitos levantados

Como mencionado na secção 2.1, após o levantamento dos requisitos, dividimo-los em três tipos de requisitos pré-definidos. Os requisitos de descrição definem a forma como o sistema se deve comportar em termos de funcionalidades (como a adição/edição/remoção de artigos do *stock* existente, permitir a realização de transações, entre outras). Os requisitos de exploração descrevem a forma como os utilizadores interagem com o sistema. Já os requisitos de controlo indicam as restrições que o sistema deve fazer cumprir.

Requisitos de descrição:

- RD1 – Cada jogo deve ser registado com um identificador único (número sequencial).
- RD2 – Ao registar um jogo no sistema devem ser ainda fornecidas as seguintes informações: nome, género, rating fornecido pela ESRB, plataforma, publicadora, desenvolvedor, ano de lançamento, quantidade em stock e preço.
- RD3 – Pode haver jogos com nomes repetidos desde que as plataformas sejam diferentes.
- RD4 – O campo “rating ESRB” deve ser preenchido com uma das seguintes categorias: “Everyone”, “10+”, “Teen”, “17+”, “18+”, “Pending”.
- RD5 – O ano de lançamento de um jogo não pode exceder o ano atual.
- RD6 – O ano de lançamento, o desenvolvedor e a publicadora de um jogo são campos de preenchimento opcional.

- RD7 – O preço de um jogo é um número decimal não menor do que 1.00. Qualquer tentativa de registo de um jogo com um preço menor do que esse valor deve ser considerado inválido pelo sistema e rejeitado.
- RD8 – Aquando da realização de uma transação (compra ou venda) de um ou mais jogos faz-se a atualização da quantidade em stock do(s) jogo(s) transacionado(s).
- RD9 – Só é possível realizar uma venda de um determinado jogo se a quantidade em stock desse mesmo jogo for igual ou superior à quantidade pretendida pelo cliente nessa transação.
- RD10 – Caso um jogo vendido por um cliente à loja ainda não esteja registado no sistema, faz-se o registo desse jogo.
- RD11 – Cada transação deve ser registada com um número de transação único e sequencial.
- RD12 – Cada transação tem associados um valor total da transação (número decimal), e uma quantidade total de jogos transacionados (número inteiro) que devem ambos ser maiores que 0.
- RD13 – Cada registo de transação tem ainda uma data e um tipo. Este último é composto por uma e uma só letra (inicialmente 'C' para Compras e 'V' para Vendas, podendo ser este domínio expandido posteriormente).
- RD14 – Em cada operação (seja compra ou venda) são transacionados 1 ou mais jogos, sendo, para cada jogo único, registados a quantidade de cópias a ser transacionadas (inteiro) e o valor total (decimal) associado a essa quantidade, ambos maiores que 0.
- RD15 – Cada transação é realizada por um cliente, que pode levar até à caixa os artigos que deseja comprar (retirados das estantes) ou vender (trazidos pelo próprio cliente). Essa transação é processada por um colaborador.
- RD16 – Para um cliente poder realizar uma transação, ele precisa estar registado no sistema da loja, sendo identificado por um número identificador único e sequencial. Caso não esteja ainda registado, será realizado o registo do cliente antes do processamento da compra.
- RD17 – Cada registo de cliente tem ainda campos obrigatórios relativos ao seu nome e data de nascimento e campos opcionais "Contacto" e "Email" que podem ter ambos mais do que um valor.
- RD18 – Também o colaborador precisa estar registado no Sistema, através do seu identificador único e do seu nome.

Requisitos de exploração:

- RE1 - Deve ser possível, a qualquer momento, obter a lista de jogos existentes na loja, ordenados crescentemente pelo seu número identificador único.
- RE2 – Deve ser permitida a pesquisa de jogos por nome, género, rating ESRB, plataforma, publicadora, desenvolvedor e ano de lançamento.
- RE3 – Deve ser possível obter uma lista com as transações realizadas numa certa data (ou em um certo espaço de tempo), ordenadas crescentemente pelo seu número de transação único.
- RE4 – No final do dia, deve ser apresentada uma lista dos jogos adicionados e removidos nesse mesmo dia, o número de registos de novos clientes e um relatório com os lucros e outras informações relativas à gestão de negócio para visualização do Sr. Clamídeo.
- RE5 – Deve ser permitida a pesquisa de transações por número de transação, por data, tipo (Compra ou Venda), cliente ou colaborador.
- RE6 – Deve ser permitida a pesquisa de clientes por número identificador de cliente, nome, contacto telefónico ou email.
- RE7 – Deve ser permitida a visualização da lista de jogos transacionados por um cliente específico, através do seu número identificador de cliente.
- RE8 – Deve ser permitida a visualização das transações processadas por um colaborador específico, através do seu número identificador de colaborador.
- RE9 – Deve ser permitida a visualização dos jogos mais vendidos em determinado período.
- RE10 – Deve ser permitida a visualização dos clientes que mais jogos compraram e venderam em determinado período.
- RE11 – Deve ser permitida a visualização do valor total de vendas e compras em determinado período.
- RE12 – Deve ser permitida a adição, atualização e remoção de jogos, clientes e colaboradores.

Requisitos de controlo:

- RC1 – Os dados relativos a clientes e transações só podem ser acedidos pelos Colaboradores. Os clientes só podem visualizar os dados relativos aos jogos e às transações que eles mesmo realizaram.
- RC2 – A adição, alteração ou remoção de jogos, clientes e transações só podem ser efetuadas por Colaboradores.
- RC3 – O acesso às informações relativas à área de negócios é restrito ao Sr. Clamídeo.

2.3. Análise e validação geral dos requisitos

Após uma revisão dos requisitos levantados e organizados por parte da equipa de trabalho, procedeu-se então à realização de uma reunião com o Sr. Clamídeo, para a qual ele levou algumas perguntas para as quais, com a análise dos requisitos, desejava obter respostas positivas:

1. Os requisitos definidos são necessários e suficientes para atender às necessidades do sistema?
2. Os requisitos são claros e compreensíveis?
3. Os requisitos não são contraditórios?
4. Os requisitos são factíveis e realistas?
5. Os requisitos estão bem definidos e não deixam margem para interpretação?
6. Existe alguma redundância em certos requisitos? (Se sim, quais?)
7. Os requisitos estão completos e não faltam requisitos importantes?
8. Os requisitos podem ser validados e verificados para garantir que o sistema atende aos objetivos definidos?
9. Foram considerados fatores externos que possam influenciar o sistema, como restrições de hardware ou software, requisitos legais, entre outros?
10. Os requisitos são relevantes para as partes interessadas envolvidas no projeto?

Após a exposição da lista de requisitos – com o formato apresentado na secção 2.2 – e a análise detalhada por parte do Sr. Clamídeo, com a ajuda e explicação da equipa de trabalho, chegou-se à conclusão de que os requisitos definidos são claros, completos e executáveis. Não foram identificadas inconsistências ou contradições nos requisitos, tornando-os adequados para atender às necessidades do sistema. Assim, foram validados os requisitos permitindo o início atempado da Fase 3 onde se irá proceder à Modelação Conceptual.

3. Modelação Conceptual

Após a definição dos requisitos e a realização de uma análise detalhada dos mesmos na Fase 2, a próxima etapa do projeto consiste na modelação conceptual do Sistema de Bases de Dados da loja de videojogos do Sr. Clamídeo. O Modelo Conceptual será construído a partir da identificação e caracterização das entidades, relacionamentos e atributos necessários para o funcionamento da loja, utilizando os requisitos previamente definidos como ponto de partida. A modelação conceptual é fundamental para o sucesso do projeto porque proporciona uma representação clara e consistente do modelo de informação da loja, permitindo uma implementação mais eficiente do SBD.

3.1. Apresentação da abordagem de modelação realizada

A modelação conceptual do Sistema de Bases de Dados da loja de videojogos do Sr. Clamídeo foi desenvolvida com base na metodologia de Connolly & Begg, utilizando um modelo Entidade-Relacionamento (ER) para representar as entidades, os relacionamentos e os atributos envolvidos no sistema. Para a representação gráfica do modelo ER, foi utilizada a notação CHEN que permite descrever as estruturas de dados de forma clara e concisa. A modelação foi realizada recorrendo ao BRModelo.

Em relação ao procedimento adotado, inicialmente identificamos os tipos de entidades e de relacionamentos relevantes para o sistema ao analisar os requisitos e a definição do sistema. Através da exploração dos mesmos, conseguimos ainda nomear atributos, associando-os a entidades e/ou relacionamentos. Estipulamos também os domínios dos atributos, especificando os tipos dos atributos e os intervalos válidos de valores que podem ser atribuídos a cada um deles.

Antes de passarmos à construção do modelo conceptual, nomeamos ainda as chaves candidatas e escolhemos a chave primária de cada entidade. Para isto ponderamos sobre quais os atributos ou combinações de atributos que poderiam ser utilizados de modo a garantir a identificação única de cada instância.

Tendo como base todas as etapas anteriores, procedemos à construção do modelo conceptual, esboçando e descrevendo as entidades, relacionamentos e atributos. Representamos visualmente as entidades como retângulos, os relacionamentos como losangos e os atributos como círculos.

Tendo o modelo conceptual edificado, revimos minuciosamente o mesmo de modo a descobrir eventuais redundâncias, inconsistências e/ou faltas de informação. Durante a revisão, analisamos cuidadosamente cada entidade, relacionamento e atributo visando garantir a sua consistência e

adequação aos requisitos do sistema. Realizamos pequenos ajustes a fim de tornar o modelo mais preciso e eficiente. Por fim, compartilhamos o modelo conceptual revisto com o Sr. Clamídeo de modo a obter o seu feedback e validação.

Nas secções que se seguem, apresentaremos o processo de desenvolvimento do modelo conceptual do sistema, bem como o modelo final.

3.2. Identificação e caracterização das entidades

A primeira etapa do desenvolvimento do Modelo Conceptual passou por definir os tipos de entidades existentes no contexto do projeto. A análise dos requisitos levantados permitiu-nos identificar as seguintes entidades:

- **Jogo** – Representa os jogos que são vendidos/comprados na loja. Tem como atributos um identificador único, nome, género, rating, plataforma, a publicadora, o desenvolvedor, ano de lançamento, o preço unitário e a sua quantidade em stock. Também se pode referir a um jogo como “Artigo” ou “Produto”, uma vez que são os únicos artigos que a loja vende. [Entidade identificada e caracterizada nos requisitos RD1, RD2, RD3, RD4, RD5, RD6, RD7, RD8, RD9, RD10, RD14, RE1, RE2, RE4, RE7, RE9, RE12, RC2]
- **Cliente** – Compra/vende um ou mais jogos à loja. Neste contexto é importante cada cliente ter uma ficha de registo na loja com os seguintes campos devidamente preenchidos: número identificador único, nome, data de nascimento, contacto telefónico e endereço de email. [RD15, RD16, RD17, RE4, RE5, RE6, RE7, RE10, RE12, RC1]
- **Colaborador** – Processa as compras/vendas aos clientes. Deve ter associados um identificador único e um nome. [RD15, RD18, RE5, RE8, RE12, RC1, RC2, RC3]
- **Transação** – Representa uma transação de um ou mais jogos entre cliente e loja. A transação é processada por um colaborador e é realizada pelo cliente e pode ser ou uma venda ou uma compra, dependendo da intenção do cliente. Para tal, esta entidade vai precisar de um atributo que indique se a transação é uma compra ou venda. Poderíamos utilizar um atributo booleano denominado, por exemplo, “isCompra” que seria True em caso de a transação ser uma compra e seria False em caso da transação ser uma venda, no entanto preferimos utilizar um CHAR de tamanho 1 que indica explicitamente o tipo de transação – C para Compra e V para Venda. Esta escolha deveu-se ao facto de tornar a entidade mais clara e a que caso surjam novos tipo de transação no futuro – empréstimos por exemplo – seja mais fácil adicionar essas novas opções. Outros atributos associados a esta entidade são o número da transação, a data da compra, a quantidade total de jogos comprados e o preço total da compra. [RD8, RD9, RD10, RD11, RD12, RD13, RD14, RD15, RE3, RE5, RE7, RE8, RE11, RC1]

Tendo identificado as entidades necessárias para a construção do modelo conceptual, podemos então organizá-las na seguinte tabela:

Designação	Descrição	Outras designações	Ocorrência
Jogo	Representa os jogos que são vendidos/comprados na loja.	Artigo, Produto.	Todos os jogos estão registados na base de dados de uma forma única.
Cliente	São os indivíduos que compram/vendem jogos na loja.	Freguês, Consumidor.	Cada cliente pode, ou não, estar registado na loja tendo um número de cliente associado em caso afirmativo.

Colaborador	Processa as transações de jogos com os clientes.	Funcionário.	Cada colaborador tem um identificador único e um nome.
Transação	Representa uma transação de um ou mais jogos entre cliente e loja. A transação é processada por um colaborador e é realizada pelo cliente. Pode ser uma venda ou uma compra.	Operação, Compra, Venda.	Cada transação tem um tipo (CHAR): C (Compra) ou V (Venda). Tem ainda associados um número sequencial único, a data em que foi efetuada, a quantidade total de jogos transacionados e o preço total respectivo.

Tabela 2 - Entidades

3.3. Identificação e caracterização dos relacionamentos

Estando as entidades definidas e continuando a ter como base os requisitos que foram levantados na Fase 2, passamos então à identificação e caracterização dos relacionamentos entre essas mesmas entidades:

Transação-(contém)-Jogo

Uma transação contém sempre pelo menos um jogo, podendo ter mais jogos. Como cada jogo pode também estar presente em uma ou mais transações, este é um relacionamento N:M no qual se deve detalhar, para cada um desses jogos, a quantidade a ser transacionada – nunca menor que 1 - e o preço total relativo a essa quantidade - calculado com base no preço unitário do jogo à data da operação.

Cliente-(realiza)-Transação

Cada cliente pode realizar várias transações, sendo que cada transação está sempre associada a um só cliente. Um cliente pode não estar registado na loja e pode não ter nenhuma transação realizada.

Transação-(processada por)-Colaborador

Cada transação é processada por um só colaborador, que por sua vez, pode processar diversas transações, pelo que este é um relacionamento 1:N. O colaborador deve associar à transação os jogos apresentados ou requeridos pelo cliente através dos IDs únicos dos jogos e definir as quantidades a serem transacionadas para cada jogo. O colaborador deve ainda processar o pagamento do valor total associado à operação e outras ações relacionadas com o processo de compra ou venda. A participação de ambas entidades neste relacionamento é obrigatória.

Entidade	Relacionamento	Cardinalidade	Participação	Entidade
Transação	contém	N:M	T:P	Jogo
Cliente	realiza	1:N	T:P	Transação
Transação	Processada por	N:1	T:T	Colaborador

Tabela 3 – Relacionamentos

3.4. Identificação e caracterização da associação dos atributos com as entidades e relacionamentos

Nesta secção iremos definir e apresentar os atributos relevantes para a descrição das entidades e dos relacionamentos que vão compor o modelo conceptual do nosso sistema. A identificação destes atributos foi feita, tal como no caso das entidades e relacionamentos, através da análise dos requisitos levantados na fase anterior. Os sete primeiros requisitos de descrição, por exemplo, sugerem e caracterizam os atributos principais da entidade “Jogo”.

Entidade/ Relacionamento	Atributo	Descrição	Tipo de Dados (Tamanho)	Nulo	Multi- valor
Jogo	ID_Jogo	Identificador único do jogo	INT	Não	Não
	Nome	Nome do jogo	VARCHAR(50)	Não	Não
	Género	Género do jogo	VARCHAR(20)	Não	Sim
	ESRB	Classificação etária atribuída pelo sistema ESRB	VARCHAR(8)	Sim	Não
	Plataforma	Plataforma que suporta o jogo	VARCHAR(20)	Não	Não
	Publicadora	Nome da empresa que publicou o jogo	VARCHAR(30)	Sim	Não
	Desenvolvedor	Nome da empresa que desenvolveu o jogo	VARCHAR(30)	Sim	Sim
	Ano_Lanc	Ano de lançamento do jogo	INT	Sim	Não
	Preço	Preço unitário do jogo (em euros)	DECIMAL(5,2)	Não	Não
	Qtd	Quantidade de cópias em stock do jogo	INT	Não	Não
Cliente	ID_Cliente	Identificador único do cliente	INT	Não	Não
	Nome	Nome do cliente	VARCHAR(50)	Não	Não
	Data_Nasc	Data de nascimento do cliente	DATE	Não	Não
	Contacto	Lista de número de telemóvel e/ou telefone	INT	Sim	Sim
	Email	Endereço email	VARCHAR(50)	Sim	Sim
Colaborador	ID_Colab	Identificador único do colaborador	INT	Não	Não
	Nome	Nome do colaborador	VARCHAR(50)	Não	Não
Transação	Num_Transac	Número sequencial único da transação	INT	Não	Não
	Data	Data em que foi efetuada a transação	DATE	Não	Não
	Qtd	Quantidade total de jogos comprados ou vendidos na transação	INT	Não	Não
	Valor_Total	Valor total da transação (em euros)	DECIMAL(5,2)	Não	Não
	Tipo	Indica se a transação é uma compra ou uma venda	CHAR(1) (C ou V)	Não	Não
Transação -(contém)- Jogo	Qtd	Quantidade de cada jogo a ser transacionada	INT	Não	Não
	Valor_Total	Valor total da transação de cada jogo (em euros e à data da transação)	DECIMAL(5,2)	Não	Não

Tabela 4 – Atributos

Tendo como objetivo a identificação única de cada uma das entidades, é ainda necessário identificar as chaves candidatas a chave primária e, de entre essas, escolher a chave primária.

Para a entidade “Jogo” as chaves candidatas que identificamos foram o atributo ID_Jogo e a combinação de atributos {Nome, Plataforma}, uma vez que são chaves que garantem a unicidade, mantendo também a simplicidade. A chave primária escolhida foi o ID_Jogo, devido a ser um atributo simples que não requer a combinação com informações adicionais, o que simplifica a modelação e torna a manipulação dos dados mais eficiente. Ao utilizar o número identificador único do jogo como chave primária, asseguramos ainda que esta nunca necessitará ser alterada, o que poderia ocorrer com o nome do jogo e/ou plataforma.

A única chave primária possível para a entidade Cliente é o seu identificador único. Apesar de a combinação entre os atributos Nome e Contacto/Email poder ser uma chave candidata, o facto de os atributos Contacto e Email serem opcionais e multivalorados torna essa combinação impraticável. O mesmo ocorre com as entidades Colaborador e Transação, com as únicas chaves candidatas a serem os respetivos número identificadores únicos, pelo que as mesmas se assumem como chaves primárias.

3.5. Apresentação e explicação do diagrama ER produzido

Tendo identificado e definido todos os constituintes do modelo conceptual passámos então à sua construção, obtendo no final o seguinte diagrama:

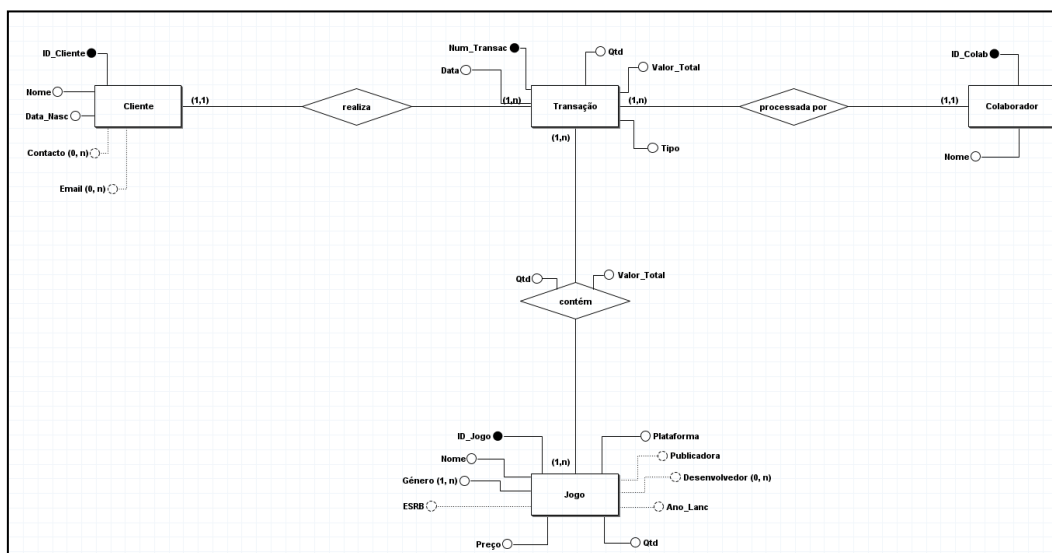


Figura 3 - Diagrama ER

O diagrama cobre todos os requisitos levantados, apresentando claramente todas as entidades - representadas por retângulos - e relacionamentos - representados por losangos - e as associações entre si, representadas por linhas. Os atributos – representados por círculos – são também associados às entidades e ao – único – relacionamento N:M também através de linhas. Usamos um círculo preenchido (preto) para representar os atributos que correspondem às chaves primárias de cada entidade.

4. Modelação Lógica

Tendo como base o modelo conceptual, e estando o mesmo já validado, podemos proceder à construção de um modelo lógico que represente a estrutura e os relacionamentos dos dados de forma mais precisa e detalhada. A criação do modelo lógico passou inicialmente pela derivação de relações que sejam capazes de representar as entidades, relacionamentos e atributos que constituem o ambiente do nosso trabalho. Além disso, aplicamos a normalização dos dados de forma a eliminar redundâncias e garantir a integridade e consistência dos dados. Finalmente, construímos o modelo lógico e procedemos a validá-lo em relação àquilo que poderão ser interrogações plausíveis no contexto do nosso sistema.

4.1. Construção e validação do modelo de dados lógico

De forma a traduzir o modelo conceptual num modelo lógico, achamos necessário iniciar este processo transitivo com a derivação de todas as relações capazes de representar as entidades, relacionamentos e atributos que constituem o modelo conceptual. Isto foi conseguido através de uma análise faseada do mesmo, que nos permitiu identificar certas estruturas e derivar a sua representação no modelo lógico. Na definição de uma relação especificamos o seu nome, os seus atributos simples e a sua chave primária e/ou chaves estrangeiras.

Primeiramente, e tendo em conta que todas as entidades que constituem o nosso modelo conceptual são entidades fortes, criamos uma relação para cada uma das entidades, incluindo todos os seus atributos simples e identificando a sua chave primária.

- **Jogo** (ID_Jogo, Nome, ESRB (opcional), Plataforma, Publicadora (opcional), Ano_Lanc (opcional), Preço, Qtd)
Chave Primária – ID_Jogo
- **Transação** (Num_Transac, Data, Qtd, Valor_Total, Tipo)
Chave Primária – Num_Transac
- **Cliente** (ID_Cliente, Nome, Data_Nasc)
Chave Primária – ID_Cliente
- **Colaborador** (ID_Colab, Nome)
Chave Primária – ID_Colab

Em seguida, para cada um dos dois relacionamentos binários 1:N, incluímos na tabela correspondente à entidade que se encontra no lado de multiplicidade N desse relacionamento uma chave estrangeira com referência à tabela da entidade no lado de multiplicidade 1. Neste caso, foi apenas atualizada a relação Transação:

- **Transação** (Num_Transac, Data, Qtd, Valor_Total, Tipo, ID_Cliente, ID_Colab)
Chave Primária – Num_Transac
Chaves Estrangeiras – ID_Cliente (Cliente) e ID_Colab (Colaborador)

Restando apenas um relacionamento, e tendo ele multiplicidade N:M, criamos uma relação com os atributos simples do relacionamento e cuja chave primária é um par composto pelas duas chaves estrangeiras, que referenciam as duas tabelas correspondentes às entidades envolvidas no relacionamento.

- **TransaçãoContem** (Num_Transac, ID_Jogo, Qtd, Valor_Total)
Chave Primária – Num_Transac, ID_Jogo
Chaves Estrangeiras – Num_Transac (Transação) e ID_Jogo (Jogo)

Finalmente, para representar cada atributo multi-valor de cada entidade, criamos uma relação que inclui a chave primária da entidade correspondente, que atua como chave estrangeira e parte da chave primária da própria relação.

- **Jogo_Gêneros** (ID_Jogo, Género)
Chave Primária – ID_Jogo, Género
Chave Estrangeira – ID_Jogo (Jogo)
- **Jogo_Desenvolvedores** (ID_Jogo, Desenvolvedor)
Chave Primária – ID_Jogo, Desenvolvedor
Chave Estrangeira – ID_Jogo (Jogo)
- **ClienteEmail** (ID_Cliente, Email)
Chave Primária – ID_Cliente, Email
Chave Estrangeira – ID_Cliente
- **ClienteContacto** (ID_Cliente, Contacto)
Chave Primária – ID_Cliente, Contacto
Chave Estrangeira – ID_Cliente

4.2. Normalização de Dados

Estando as relações identificadas, é ainda necessário eliminar redundâncias e anomalias nos dados de forma a aumentar a integridade dos dados e a eficiência de armazenamento. Para isso recorreremos ao processo de normalização de dados, que envolve a aplicação de um conjunto de regras - denominadas “Formas Normais” – que definem os critérios para organizar os dados de forma estruturada e sem ambiguidades.

A 1ª Forma Normal estabelece os requisitos básicos para a estrutura de uma relação em um modelo de bases de dados relacional, exigindo que o domínio de cada atributo contenha apenas valores atômicos e que cada atributo contenha um único valor do seu domínio. Como podemos verificar ao consultar as relações estabelecidas na secção anterior, o nosso modelo já atende aos requisitos desta primeira forma normal, uma vez que cada atributo possui um único valor e não há atributos compostos.

A 2ª Forma Normal define que numa relação, nenhum atributo não-primo deve ser funcionalmente dependente de um subconjunto próprio de qualquer chave candidata, ou seja, todos os atributos não-primos devem ser completamente dependentes da chave primária.

A 3ª Forma Normal estipula que nenhuma relação pode ter dependências transitivas, ou seja que nenhum atributo não-primo deve ser dependente de outro atributo não-primo.

Como já havíamos concluído para a 1ª Forma Normal, também para as segunda e terceira formas podemos observar que o nosso modelo está em conformidade com as mesmas, pelo que concluímos que o modelo lógico está normalizado até à terceira forma normal.

4.3. Apresentação e explicação do modelo lógico produzido

Após a derivação de todas as relações necessárias e normalização dos dados presentes nas mesmas, passamos então à construção do modelo lógico no *MySQL Workbench*. Criámos então nove tabelas de acordo com a derivação efetuada na secção 4.1 – “Transação”, “Colaborador”, “Cliente”, “ClienteEmail”, “ClienteContacto”, “TransaçãoContem”, “Jogo”, “Jogo_Géneros” e “Jogo_Desenvolvedores”. Em cada uma das tabelas, inserimos os seus atributos e definimos os domínios dos mesmos. Definimos também a chave primária, selecionando o(s) atributo(s) que a forma(m), e a(s) chave(s) estrangeira(s). Conectamos ainda as tabelas, sendo que os relacionamentos representados por linhas a tracejado representam relacionamentos não-obrigatórios, ou seja, indicam a possibilidade de que um registo de uma tabela esteja relacionado a um ou mais registos de outra tabela, ao contrário dos relacionamentos representados por linhas “cheias” que representam a existência de uma restrição de integridade referencial, onde a chave primária de uma tabela é referenciada como chave estrangeira em outra tabela.

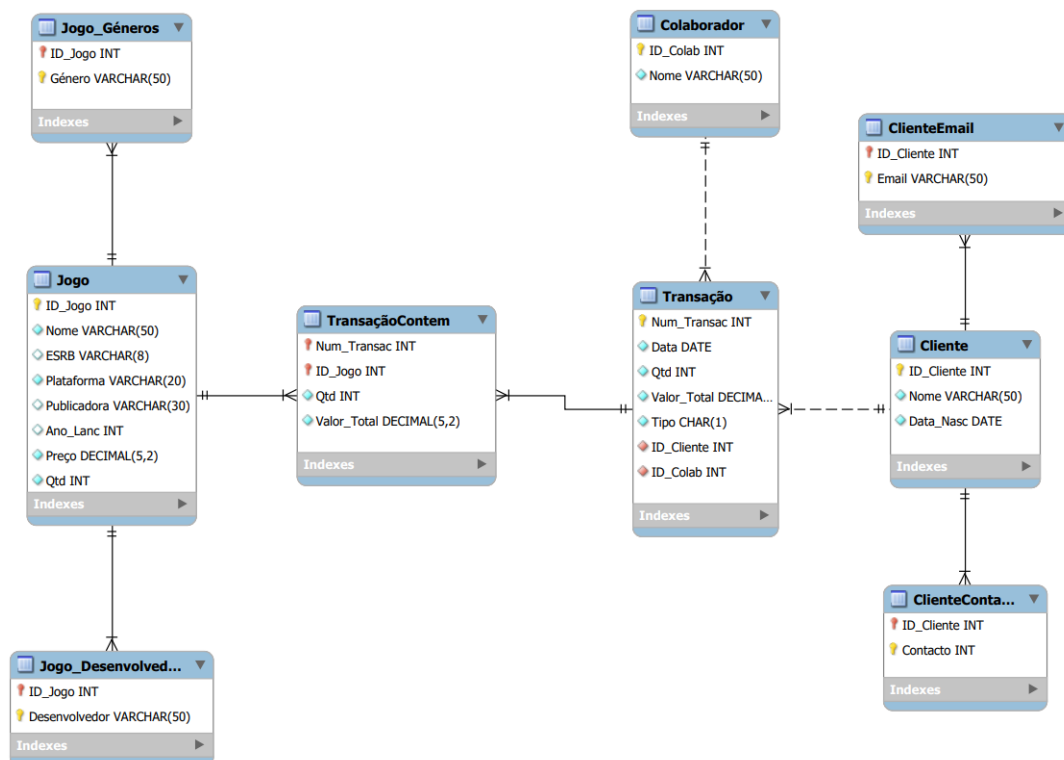


Figura 4 - Modelo Lógico

4.4. Validação do modelo com interrogações do utilizador

Para concluir a fase de modelação lógica, resta validar o modelo construído através de possíveis interrogações do utilizador. Através dessas interrogações examinaremos a capacidade do modelo de atender às necessidades e consultas dos utilizadores.

- **Posso obter uma lista de todos os jogos em stock na loja com todas as informações associadas a cada jogo?**

Para obter a lista de todos os jogos em stock na loja basta consultar a tabela “Jogo” e seleccionar todas as colunas, filtrando os jogos pelos que têm quantidade (de stock em loja) maior do que zero.

- **Posso obter a quantidade e o valor total transacionados de um determinado jogo em uma determinada transação?**

Filtrando a pesquisa pelo número de transação e o identificador do jogo, podemos procurar na tabela “TransaçãoContem” e obter esses valores. Se tivermos apenas o nome do jogo, poderemos ter de procurar previamente os ID_Jogo’s na tabela “Jogo” associados a esse nome.

- **É possível obter a média dos preços dos jogos que foram lançados num dado ano para uma data plataforma?**

Para fazer isso, é preciso consultar a tabela “Jogo” e utilizar uma cláusula de filtro para seleccionar apenas os jogos com o ano de lançamento e plataforma desejados, aplicando também a função agregada AVG aos Preços para obter a média dos preços.

- **Posso obter os contactos de um cliente através do seu ID?**

Sim, realizando uma consulta na tabela "ClienteContacto" relacionada com a tabela "Cliente", utilizando o ID do cliente como critério de pesquisa.

- **É possível obter a lista de jogos (com nome, plataforma e quantidade de transação) transacionados por um dado cliente?**

Para obter o pretendido, primeiramente seleccionamos da tabela “Transação” apenas as transações que foram efetuadas pelo cliente (representado pelo seu identificador único) passando depois a obter da tabela “TransaçãoContem” os identificadores dos jogos associados aos números de transações que representam as transações efetuadas pelo dado cliente. Com esses IDs, obtemos da tabela “Jogo” os nomes e as plataformas. Também retiramos a informação relativa às quantidades transacionadas da tabela “TransaçãoContem”.

- **Quais foram as transações realizadas por um determinado colaborador em uma data específica?**

Verificamos a tabela “Transação” para identificar as transações associadas ao identificador único do colaborador realizadas na data pretendida.

- **Como obtemos os nomes dos clientes que compraram jogos cujo preço é maior do que um dado valor?**

Na nossa procura, definimos o limite mínimo para o preço dos jogos, obtendo da tabela Jogo os identificadores de apenas aqueles que cumprem esse critério. Com os identificadores dos jogos,

obtemos da tabela “TransaçãoContem” os números das transações associados a esses jogos, usando esses números como filtro para adquirir da tabela “Transação” todos os identificadores de clientes associados, filtrando também apenas as Transações do tipo ‘C’ (compra). Por fim, obtemos os nomes dos Clientes da tabela “Cliente”.

- **Como obtemos o histórico de transações de um cliente num dado intervalo de tempo, por ordem do mais recente para o mais antigo?**

Filtramos a tabela “Transação” com base no ID do cliente e nas datas de início e fim fornecidas, ordenando os resultados pela data da transação em ordem decrescente, mostrando as transações mais recentes primeiro.

As respostas às interrogações demonstram que o modelo lógico construído deverá ser capaz de fornecer as informações solicitadas de forma eficiente. Através das consultas pretendidas, verificamos que o modelo lógico atende aos requisitos do sistema, permitindo a extração de dados relevantes para diversos tipos de consultas.

Findo o desenvolvimento do modelo lógico e respetiva validação, temos a confirmação de que o modelo está bem construído e pronto para ser utilizado como base na próxima fase do processo de desenvolvimento do nosso sistema.

5. Implementação Física

Com a validação do modelo lógico produzido na fase anterior, chegamos à etapa na qual faremos a transição da parte mais “abstrata” (ou teórica) para a implementação mais prática do nosso Sistema de Bases de Dados, colocando em prática todos os conceitos e estruturas definidos anteriormente, dando vida ao projeto.

O início da implementação física é o momento em que colocamos “as mãos na massa” e começamos a construir realmente o nosso sistema, traduzindo o modelo lógico para o sistema de gestão de bases de dados escolhido, utilizando a linguagem SQL como ferramenta principal. Para além disso, traduzimos algumas interrogações do utilizador – também apresentadas na fase anterior – para SQL, definimos e caracterizamos as vistas de utilização em SQL, calculamos o espaço da base de dados inicial e a sua taxa de crescimento anual e indexamos o sistema de dados, apresentando por fim os procedimentos implementados e o plano de segurança e recuperação de dados.

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

Nesta primeira secção, utilizando linguagem SQL, criamos as tabelas com as respetivas colunas e tipos de dados, além de definir a chave primária e, se necessário, chave(s) estrangeira(s) de cada uma delas. Essas informações foram previamente estabelecidas durante a fase de modelação lógica, o que simplificou e facilitou a tradução para SQL, deixando apenas como nota uma pequena alteração na tradução do nome do atributo “Data” da relação “Transação” para a coluna “Data_Transac” da tabela “Transação”, o que se deveu a “Data” ser uma palavra reservada em SQL.

De seguida, apresentaremos os excertos do script que demonstram a criação de todas as tabelas necessárias, juntamente com todas as suas características mencionadas.

```
CREATE TABLE IF NOT EXISTS Jogo (
    ID_Jogo INT,
    Nome VARCHAR(50) NOT NULL,
    ESRB VARCHAR(8) NULL,
    Plataforma VARCHAR(20) NOT NULL,
    Publicadora VARCHAR(30) NULL,
    Ano_Lanc INT NULL,
    Preço DECIMAL(5,2) NOT NULL,
    Qtd INT NOT NULL,
    PRIMARY KEY (ID_Jogo)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;
```

Figura 5 - Criação da tabela Jogo

```
CREATE TABLE IF NOT EXISTS Cliente (
    ID_Cliente INT NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    Data_Nasc DATE NOT NULL,
    PRIMARY KEY (ID_Cliente)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;
```

Figura 6- Criação da tabela Cliente

```
CREATE TABLE IF NOT EXISTS Colaborador (
    ID_Colab INT NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID_Colab)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;
```

Figura 7 - Criação da tabela Colaborador


```

CREATE TABLE IF NOT EXISTS Transação (
    Num_Transac INT NOT NULL,
    Data_Transac DATE NOT NULL,
    Qtd INT NOT NULL,
    Valor_Total DECIMAL(5,2) NOT NULL,
    Tipo CHAR(1) NOT NULL,
    ID_Cliente INT NOT NULL,
    ID_Colab INT NOT NULL,
    PRIMARY KEY (Num_Transac),
    FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente),
    FOREIGN KEY (ID_Colab) REFERENCES Colaborador(ID_Colab)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

```

Figura 8 - Criação da tabela Transação

```

CREATE TABLE IF NOT EXISTS TransaçãoContem (
    Num_Transac INT NOT NULL,
    ID_Jogo INT NOT NULL,
    Qtd INT NOT NULL,
    Valor_Total DECIMAL(5,2) NOT NULL,
    PRIMARY KEY (Num_Transac, ID_Jogo),
    FOREIGN KEY (Num_Transac) REFERENCES Transação(Num_Transac),
    FOREIGN KEY (ID_Jogo) REFERENCES Jogo(ID_Jogo)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

```

Figura 9 - Criação da tabela TransaçãoContem

```

CREATE TABLE IF NOT EXISTS Jogo_Gêneros (
    ID_Jogo INT NOT NULL,
    Gênero VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID_Jogo, Gênero),
    FOREIGN KEY (ID_Jogo) REFERENCES Jogo(ID_Jogo)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS Jogo_Desenvolvedores (
    ID_Jogo INT NOT NULL,
    Desenvolvedor VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID_Jogo, Desenvolvedor),
    FOREIGN KEY (ID_Jogo) REFERENCES Jogo(ID_Jogo)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS ClienteEmail (
    ID_Cliente INT NOT NULL,
    Email VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID_Cliente, Email),
    FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS ClienteContacto (
    ID_Cliente INT NOT NULL,
    Contacto INT NOT NULL,
    PRIMARY KEY (ID_Cliente, Contacto),
    FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente)
)

ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4;

```

Figura 10 - Criação das tabelas referentes às relações criadas para atributos multi-valor

5.2. Tradução das interrogações do utilizador para SQL

Nesta secção iremos apresentar alguns exemplos de código SQL que permitem traduzir as respostas às interrogações de utilizador levantadas na secção 4.4.

- **Obter uma lista de todos os jogos em stock na loja com todas as informações associadas a cada jogo**

```
DELIMITER $$
• CREATE PROCEDURE StockJogos()
BEGIN
    SELECT *
    FROM Jogo
    WHERE Jogo.Qtd > 0;
END $$
DELIMITER ;
```

Figura 11 - Listagem de jogos em stock

- **Consultar a quantidade transacionada e o valor total transacionado associados a um número de transação e nome do jogo dados**

```
DELIMITER $$
• CREATE PROCEDURE QtdETotalTransacionados(IN num_transac INT, IN nome VARCHAR(50))
BEGIN
    SELECT TC.Qtd, TC.Valor_Total
    FROM TransaçãoContem TC
    INNER JOIN Jogo J ON TC.ID_Jogo = J.ID_Jogo
    WHERE TC.Num_Transac = num_transac
    AND J.Nome = nome;
END $$
DELIMITER ;
```

Figura 12 - Consulta da quantidade e valor total transacionados para um dado jogo em uma determinada transação

- **Obter a média dos preços dos jogos que foram lançados em um dado ano para uma dada plataforma**

```
DELIMITER $$
CREATE FUNCTION MediaPrecosJogosAnoLancEPlataforma(ano_lanc INT, plataforma VARCHAR(20))
RETURNS FLOAT DETERMINISTIC
BEGIN
    RETURN (SELECT AVG(J.Preço)
            FROM Jogo J
            WHERE J.Ano_Lanc = ano_lanc AND J.Plataforma = plataforma);
END $$
DELIMITER ;
```

Figura 13 - Função para obter média dos preços dos jogos para ano e plataforma dados

- Obter a lista de jogos (com nome, plataforma e quantidade de transação) transacionados por um dado cliente

```

DELIMITER $$
CREATE PROCEDURE ObterListaJogosTransacionados(IN cliente_id INT)
BEGIN
    SELECT J.Nome, J.Plataforma, TC.Qtd
    FROM Jogo J
    JOIN TransaçãoContem TC ON J.ID_Jogo = TC.ID_Jogo
    JOIN Transação T ON TC.Num_Transac = T.Num_Transac
    WHERE T.ID_Cliente = cliente_id;
END $$
DELIMITER ;

```

Figura 14 - Procedimento para obter a lista de jogos transacionados por um dado cliente

- Obter nomes dos clientes que compraram jogos cujo preço é maior do que um valor dado

```

DELIMITER $$
CREATE PROCEDURE ObterClientesComJogosAcimaDoPreco(IN preco_limite DECIMAL(5,2))
BEGIN
    SELECT DISTINCT C.Nome
    FROM Cliente C
    JOIN Transação T ON C.ID_Cliente = T.ID_Cliente
    JOIN TransaçãoContem TC ON T.Num_Transac = TC.Num_Transac
    JOIN Jogo J ON TC.ID_Jogo = J.ID_Jogo
    WHERE J.Preço > preco_limite AND T.Tipo = 'V';
END $$
DELIMITER ;

```

Figura 15 - Procedimento para obter os nomes dos clientes que compraram jogos a partir de um certo preço

- Ver histórico de transações de um cliente num intervalo de tempo, por ordem do mais recente para o mais antigo

```

DELIMITER $$
CREATE PROCEDURE historico_transacao(IN ID_Cliente INT, IN Data_inicio DATETIME, IN Data_fim DATETIME)
BEGIN
    SELECT *
    FROM Transação as t
    WHERE t.ID_Cliente = ID_Cliente AND t.Data_Transac >= Data_inicio AND t.Data_Transac <= Data_fim
    ORDER BY t.Data_Transac DESC;
END $$
DELIMITER ;

```

Figura 16 - Procedimento que lista transações de um cliente num intervalo de tempo, ordenadas decrescentemente por data de transação

- **Listar as 10 transações com valor total mais elevado**

```

DELIMITER //
CREATE PROCEDURE Top10Transacoes()
BEGIN
    SELECT T.Valor_Total, T.Tipo, T.Num_Transac, C.ID_Cliente
    FROM Transação AS T INNER JOIN Cliente AS C
    ON T.ID_Cliente=C.ID_Cliente
    ORDER BY Valor_Total DESC
    LIMIT 10;
END //
DELIMITER ;

```

Figura 17 - Procedimento que lista as 10 transações de maiores valores

- **Listar os 10 jogos mais vendidos da loja**

```

DELIMITER //
CREATE PROCEDURE Top10Jogos()
BEGIN
    SELECT J.ID_Jogo, J.Nome, SUM(TC.Qtd) AS TotalQtd
    FROM Jogo AS J
    JOIN TransaçãoContem AS TC ON J.ID_Jogo = TC.ID_Jogo
    GROUP BY J.ID_Jogo, J.Nome
    ORDER BY TotalQtd DESC
    LIMIT 10;
END //
DELIMITER ;

```

Figura 18 - Procedimento que lista os 10 jogos mais vendidos de sempre da loja

5.3. Definição e caracterização das vistas de utilização em SQL

As vistas são consultas armazenadas no Sistema de Gestão de Bases de Dados que produzem resultados pré-calculados e armazenados como uma tabela virtual. No fundo, podem ser consideradas como tabelas dinâmicas resultantes de várias operações entre tabelas ou outras vistas.

Ao definir e caracterizar as vistas apresentadas em seguida, demonstramos como as consultas SQL podem ser encapsuladas em vistas para simplificar o acesso aos dados e melhorar a eficiência das operações de consulta.

- **Vista de resumo de transações por cliente** - Retorna um resumo das transações realizadas por cada cliente (representado pelo seu ID e pelo seu nome), incluindo o número total de transações, o valor total transacionado e a data da última transação.

```

CREATE VIEW TransaçõesPorCliente AS
SELECT t.ID_Cliente, c.Nome AS NomeCliente, COUNT(*) AS TotalTransacoes, SUM(t.Valor_Total) AS ValorTotal, MAX(t.Data_Transac) AS UltimaTransacao
FROM Transação AS t
JOIN Cliente AS c ON t.ID_Cliente = c.ID_Cliente
GROUP BY t.ID_Cliente;

```

Figura 19 - Vista de resumo de transações por cliente

- **Vista dos jogos mais populares** - lista os jogos mais vendidos (ID e Nome), com base no número total de unidades vendidas em todas as transações. A lista é apresentada ordenada decrescentemente.

```
CREATE VIEW JogosMaisPopulares AS
SELECT J.ID_Jogo, J.Nome, SUM(TC.Qtd) AS TotalUnidadesVendidas
FROM Jogo AS J
JOIN TransaçãoContem AS TC ON J.ID_Jogo = TC.ID_Jogo
GROUP BY J.ID_Jogo
ORDER BY TotalUnidadesVendidas DESC;
```

Figura 20 - Vista dos jogos mais populares

- **Vista de disponibilidade de stock** - mostra a quantidade disponível de cada jogo em estoque, com base nas transações de compra e venda.

```
CREATE VIEW DisponibilidadeStock AS
SELECT J.ID_Jogo, J.Nome, J.Qtd - COALESCE(TotalVendido, 0) + COALESCE(TotalComprado, 0) AS StockDisponivel
FROM Jogo AS J
LEFT JOIN (
    SELECT tc.ID_Jogo, SUM(tc.Qtd) AS TotalVendido
    FROM TransaçãoContem AS tc
    JOIN Transação ON tc.Num_Transac = Transação.Num_Transac
    WHERE Transação.Tipo = 'V'
    GROUP BY ID_Jogo
) AS Vendas ON J.ID_Jogo = Vendas.ID_Jogo
LEFT JOIN (
    SELECT tc.ID_Jogo, SUM(tc.Qtd) AS TotalComprado
    FROM TransaçãoContem AS tc
    JOIN Transação ON tc.Num_Transac = Transação.Num_Transac
    WHERE Transação.Tipo = 'C'
    GROUP BY ID_Jogo
) AS Compras ON J.ID_Jogo = Compras.ID_Jogo;
```

Figura 21 - Vista de Disponibilidade de Stock

5.4. Cálculo do espaço da Base de Dados

Nesta secção vamos fazer uma estimativa do espaço necessário para armazenar os dados iniciais do sistema e planear o crescimento futuro da base de dados. Com uma primeira estimativa, poderemos dimensionar corretamente os recursos de armazenamento, garantindo que há capacidade suficiente para armazenar os dados num estado inicial. Além disso, ao definir a taxa de crescimento anual, estabelecemos uma estimativa do aumento do tamanho da base de dados ao longo do tempo, permitindo-nos planear a capacidade do sistema no longo prazo e garantir que o ambiente de armazenamento é suficientemente escalável para acomodar o crescimento dos dados.

Começamos por determinar quanto é que cada ocorrência de cada relação (com os seus atributos) ocuparia em disco, obtendo a seguinte tabela:

Relação	Atributo	Tipo de Dados	Tamanho	Total
Jogo	ID_Jogo	INT	4 bytes	123 bytes
	Nome	VARCHAR(50)	50 bytes	
	ESRB	VARCHAR(8)	8 bytes	
	Plataforma	VARCHAR(20)	20 bytes	
	Publicadora	VARCHAR(30)	30 bytes	
	Ano_Lanc	INT	4 bytes	
	Preço	DECIMAL(5,2)	3 bytes	
	Qtd	INT	4 bytes	
Cliente	ID_Cliente	INT	4 bytes	57 bytes
	Nome	VARCHAR(50)	50 bytes	
	Data_Nasc	DATE	3 bytes	
Colaborador	ID_Colab	INT	4 bytes	54 bytes
	Nome	VARCHAR(50)	50 bytes	
Transação	Num_Transac	INT	4 bytes	20 bytes
	Data_Transac	DATE	3 bytes	
	Qtd	INT	4 bytes	
	Valor_Total	DECIMAL(5,2)	3 bytes	
	Tipo	CHAR(1)	1 byte	
	ID_Cliente	INT	4 bytes	
	ID_Colab	INT	4 bytes	
TransaçãoContem	Num_Transac	INT	4 bytes	15 bytes
	ID_Jogo	INT	4 bytes	
	Qtd	INT	4 bytes	
	Valor_Total	DECIMAL(5,2)	3 bytes	
Jogo_Gêneros	ID_Jogo	INT	4 bytes	54 bytes
	Gênero	VARCHAR(50)	50 bytes	
Jogo_Desenvolvedores	ID_Jogo	INT	4 bytes	54 bytes
	Desenvolvedor	VARCHAR(50)	50 bytes	
ClienteEmail	ID_Cliente	INT	4 bytes	54 bytes
	Email	VARCHAR(50)	50 bytes	
ClienteContacto	ID_Cliente	INT	4 bytes	8 bytes
	Contacto	INT	4 bytes	

NOTA: Assumimos que atributos do tipo DECIMAL(x,y) têm tamanho (x/2)+1

Assim sendo, e tendo um povoamento inicial com, por exemplo, 100 jogos, 500 clientes, 2 colaboradores, 1000 transações, 1500 ocorrências da relação “TransaçãoContem”, 1600 da relação “Jogo_Géneros”, 1050 da relação “Jogo_Desenvolvedores”, 1400 da relação “ClienteEmail” e 1300 da relação “ClienteContacto”, estimamos serem necessários 312.508 bytes de espaço ($123 \cdot 100 + 57 \cdot 500 + 54 \cdot 2 + 20 \cdot 1000 + 15 \cdot 1500 + 54 \cdot 1600 + 54 \cdot 1050 + 54 \cdot 1400 + 8 \cdot 1300$).

Considerando uma estimativa do aumento médio do nº de clientes de 12.5% ao ano, do nº de jogos em 5% por ano e das transações (e ocorrências de “TransaçãoContem”) em 10% ao ano, necessitaremos, ao final do 1º ano, de 330.591 bytes, o que corresponde a um aumento de 5.79% anual. Ao fim de 5 anos, necessitaríamos, com base nas nossas estimativas e nos nossos cálculos, de 2.033.545.800 bytes (~1.89GB).

5.5. Indexação do Sistema de Dados

Nesta secção iremos proceder à indexação do nosso sistema, que é uma técnica que permite acelerar as operações de procura e recuperação de dados, tornando as consultas mais eficientes. Ao criar índices em colunas específicas de certas tabelas, estamos essencialmente a criar uma estrutura adicional de dados que organiza essas colunas de forma otimizada. Estes índices são usados pelo Sistema de Gestão de Bases de Dados para localizar rapidamente os registos relevantes, em vez de efetuar uma pesquisa sequencial completa na tabela. Esta técnica é particularmente útil em situações de consultas frequentes a certas colunas específicas de uma tabela, pelo que decidimos indexar as colunas mais relevantes das tabelas mais consultadas (Jogo, Transação e Cliente), desenvolvendo o seguinte excerto de script:

```
CREATE INDEX idx_nome_plataforma ON Jogo (Nome, Plataforma);  
CREATE INDEX idx_idc_transacao ON Transação (ID_Cliente);  
CREATE INDEX idx_nome_cliente ON Cliente (Nome);
```

Figura 22 - Código SQL para a criação dos índices

5.6. Procedimentos implementados

Exploraremos agora a criação de procedimentos relevantes no contexto do nosso caso prático, tendo em consideração as necessidades e requisitos específicos do sistema. A implementação destes procedimentos visa aprimorar a eficiência e capacidade de resposta do mesmo.

O primeiro procedimento implementado é usado para registar um novo cliente na base de dados. Recebe como parâmetros o nome do cliente, a data de nascimento e duas “strings” que contêm os emails e os contactos do cliente. Obtemos o ID_Cliente somando uma unidade ao ID máximo já presente na tabela. Caso ainda não existam registos na tabela, o ID é 1. O novo Cliente é inserido na tabela com o ID calculado e os dados fornecidos, sendo que caso o cliente seja registado também com um ou mais emails e/ou um ou mais contactos (em ambos os casos cada email/contacto é separado por uma vírgula) são também inseridos novos registos nas respetivas tabelas, em que cada email e cada contacto é inserido em conjunto com o ID do cliente.

```
-- Procedimento para registar um novo cliente
DELIMITER $$

CREATE PROCEDURE RegistrarCliente(
    IN nome VARCHAR(50),
    IN data_nascimento DATE,
    IN emails VARCHAR(255),
    IN contactos VARCHAR(255)
)
BEGIN
    DECLARE novoClienteID INT;
    DECLARE email VARCHAR(50);
    DECLARE contacto INT;

    -- Obter o ID do novo cliente a inserir
    SELECT MAX(ID_Cliente) + 1 INTO novoClienteID FROM Cliente;
    IF novoClienteID IS NULL THEN
        SET novoClienteID = 1;
    END IF;

    -- Inserir o novo cliente na tabela Cliente
    INSERT INTO Cliente (ID_Cliente, Nome, Data_Nasc)
    VALUES (novoClienteID, nome, data_nascimento);

    -- Inserir os emails do cliente, se fornecidos
    WHILE LENGTH(emails) > 0 DO
        SET email = SUBSTRING_INDEX(emails, ',', 1);
        INSERT INTO ClienteEmail (ID_Cliente, Email) VALUES (novoClienteID, email);
        SET emails = SUBSTRING(emails, LENGTH(email) + 2);
    END WHILE;

    -- Inserir os contactos do cliente, se fornecidos
    WHILE LENGTH(contactos) > 0 DO
        SET contacto = CAST(SUBSTRING_INDEX(contactos, ',', 1) AS UNSIGNED);
        INSERT INTO ClienteContacto (ID_Cliente, Contacto) VALUES (novoClienteID, contacto);
        SET contactos = SUBSTRING(contactos, LENGTH(contacto) + 2);
    END WHILE;

END $$
DELIMITER ;
```

Figura 23 - Procedimento para registar um novo cliente

De similar modo, implementamos o procedimento que regista um novo jogo na base de dados.

```
-- Procedimento para registar um novo jogo
DELIMITER $$

CREATE PROCEDURE RegistrarJogo(
    IN nome VARCHAR(50),
    IN esrb VARCHAR(8),
    IN plataforma VARCHAR(20),
    IN publicadora VARCHAR(30),
    IN ano_lanc INT,
    IN preço DECIMAL(5, 2),
    IN qtd INT,
    IN generos VARCHAR(255),
    IN desenvolvedores VARCHAR(255)
)
BEGIN
    DECLARE novoJogoID INT;
    DECLARE genero VARCHAR(50);
    DECLARE desenvolvedor VARCHAR(50);

    -- Obter o ID do novo jogo a ser inserido
    SELECT MAX(ID_Jogo) + 1 INTO novoJogoID FROM Jogo;
    IF novoJogoID IS NULL THEN
        SET novoJogoID = 1;
    END IF;

    -- Inserir o novo jogo na tabela Jogo
    INSERT INTO Jogo (ID_Jogo, Nome, ESRB, Plataforma, Publicadora, Ano_Lanc, Preço, Qtd)
    VALUES (novoJogoID, nome, esrb, plataforma, publicadora, ano_lanc, preço, qtd);

    -- Inserir os géneros do jogo, se fornecidos
    WHILE LENGTH(generos) > 0 DO
        SET genero = SUBSTRING_INDEX(generos, ',', 1);
        INSERT INTO Jogo_Géneros (ID_Jogo, Género) VALUES (novoJogoID, genero);
        SET generos = SUBSTRING(generos, LENGTH(genero) + 2);
    END WHILE;

    -- Inserir os desenvolvedores do jogo, se fornecidos
    WHILE LENGTH(desenvolvedores) > 0 DO
        SET desenvolvedor = SUBSTRING_INDEX(desenvolvedores, ',', 1);
        INSERT INTO Jogo_Desenvolvedores (ID_Jogo, Desenvolvedor) VALUES (novoJogoID, desenvolvedor);
        SET desenvolvedores = SUBSTRING(desenvolvedores, LENGTH(desenvolvedor) + 2);
    END WHILE;

END $$
DELIMITER ;
```

Figura 24 - Procedimento para registar um novo jogo

Já o terceiro, e último, procedimento implementado que irá ser útil no povoamento inicial traduz o registo de uma nova transação. Começa por obter o número único da transação a registar, através do mesmo método que os procedimentos anteriores, inserindo depois o novo registo na tabela Transação. Inicialmente este registo, tem os valores “Qtd” e “Valor_Total” a zero, pois eles serão apenas no fim calculados, tendo por base as somas de todas as “Qtd” e “Valor_Total” de todos os novos registos de igual Num_Transac adicionados à tabela TransaçãoContem.

Percorremos ciclicamente a variável `jogos_qtd` – que guarda as informações (`id_jogo,qtd`) dos jogos a adicionar no formato `<id_jogo1,qtd1;id_jogo2,qtd2;(…);id_jogoN,qtdN;>` - guardando todas as informações que serão no fim de cada iteração adicionadas num novo registo à tabela `TransaçãoContem`. Para cada um dos jogos transacionados é também removida ou adicionada (dependendo do tipo de Transação ser 'V' ou 'C', respetivamente) a quantidade transacionada à “Qtd” existente em stock para o respetivo jogo na tabela “Jogo”.

Após o ciclo, são então atualizados os campos “Qtd” e “Valor_Total” da tabela `Transação` para a transação registada.

```
DELIMITER $$
CREATE PROCEDURE RegistrarTransacao(
    IN data_transac DATE,
    IN id_cliente INT,
    IN id_colab INT,
    IN jogos_qtd VARCHAR(255),
    IN tipo CHAR(1)
)
BEGIN
    DECLARE novoNumTransac INT;
    DECLARE operacao INT;

    -- Obter o novo número de transação
    SELECT MAX(Num_Transac) + 1 INTO novoNumTransac FROM Transação;
    IF novoNumTransac IS NULL THEN
        SET novoNumTransac = 1;
    END IF;

    -- Define a operação a ser realizada na tabela Jogo com base no tipo de transação
    IF tipo = 'C' THEN
        SET operacao = 1; -- Adicionar à Qtd_Disponivel
    ELSEIF tipo = 'V' THEN
        SET operacao = -1; -- Subtrair da Qtd_Disponivel
    END IF;

    -- Insere um registo inicial na tabela Transação
    INSERT INTO Transação (Num_Transac, Data_Transac, Qtd, Valor_Total, Tipo, ID_Cliente, ID_Colab)
    VALUES (novoNumTransac, data_transac, 0, 0, tipo, id_cliente, id_colab);

    -- Percorre os jogos e suas quantidades na string jogos_qtd
    WHILE LENGTH(jogos_qtd) > 0 DO
        -- Extrai o id_jogo e a qtd para o jogo atual
        SET @jogo_qtd := SUBSTRING_INDEX(jogos_qtd, ';', 1);
        SET @id_jogo := SUBSTRING_INDEX(@jogo_qtd, ',', 1);
        SET @qtd := SUBSTRING_INDEX(@jogo_qtd, ',', -1);

        -- Obtém o preço do jogo da tabela Jogo
        SET @preco := (SELECT Preço FROM Jogo WHERE id_jogo = @id_jogo);

        -- Calcula o valor total para o jogo atual
        SET @valor_total := @qtd * @preco;

        -- Insere um registo na tabela TransaçãoContem
        INSERT INTO TransaçãoContem (Num_Transac, id_jogo, qtd, valor_total)
        VALUES (novoNumTransac, @id_jogo, @qtd, @valor_total);

        -- Atualiza a quantidade disponível em stock na tabela Jogo
        UPDATE Jogo
        SET Qtd = Qtd + (@qtd * operacao)
        WHERE ID_Jogo = @id_jogo;

        -- Remove o jogo processado da string jogos_qtd
        SET jogos_qtd := REPLACE(jogos_qtd, CONCAT(@jogo_qtd, ';'), '');
    END WHILE;

    -- Calcula o qtdtotal e valor_total para a transação
    SELECT SUM(qtd) INTO @qtdtotal FROM TransaçãoContem WHERE Num_Transac = novoNumTransac;
    SELECT SUM(valor_total) INTO @valor_total FROM TransaçãoContem WHERE Num_Transac = novoNumTransac;

    -- Atualiza o registo inicial na tabela Transacao
    UPDATE Transação
    SET Qtd = @qtdtotal, Valor_Total = @valor_total
    WHERE Num_Transac = novoNumTransac;

    -- Faz o commit para salvar as alterações
    COMMIT;
END $$
DELIMITER ;
```

Figura 25 - Procedimento para registar uma Transação

5.7. Plano de segurança e recuperação de dados

De modo a garantirmos o bom funcionamento da base de dados em termos do acesso e uso às ferramentas implementadas, é necessário criar três perfis de utilização: Clamídeo, Funcionário e Utilizador.

O perfil “Clamídeo” é destinado, como o próprio nome indica, ao Sr. Clamídeo – o único administrador do sistema - e possui privilégios completos, o que lhe permite executar todas as operações na base de dados e ter acesso a todas as informações armazenadas nela, sem restrições.

```
CREATE USER 'clamídeo'@'localhost' IDENTIFIED BY 'clamídeo';  
GRANT ALL PRIVILEGES ON loja.* TO 'clamídeo'@'localhost';
```

Figura 26 - Código SQL para a criação do perfil Clamídeo

O perfil “Funcionário” é destinado ao Serginho ou a outros funcionários que eventualmente sejam contratados. O Funcionário partilha grande parte das responsabilidades do Sr. Clamídeo, no entanto não pode, logicamente, realizar operações de consulta, inserção ou remoção relacionadas com a Tabela Colaborador. Este perfil tem, então, as permissões necessárias para consultar e processar Transações e consultar e adicionar/remover Jogos e Clientes.

```
CREATE USER 'funcionario'@'localhost' IDENTIFIED BY 'funcionario';  
GRANT SELECT, INSERT, UPDATE ON loja.Cliente TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.ClienteEmail TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.ClienteContacto TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.Transacção TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.TransacçãoContem TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.Jogo TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.Jogo_Géneros TO 'funcionario'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.Jogo_Desenvolvedores TO 'funcionario'@'localhost';
```

Figura 27 - Código SQL para a criação do perfil Funcionário

Por fim, o “Utilizador” pode efetuar, consultar e alterar a sua ficha de registo, visualizar os jogos existentes, bem como os respetivos Géneros e Desenvolvedores. Para além disso, pode ainda consultar um conjunto de atributos da tabela Transacção.

```
CREATE USER 'utilizador'@'localhost' IDENTIFIED BY 'utilizador';  
GRANT SELECT, INSERT, UPDATE ON loja.Cliente TO 'utilizador'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.ClienteEmail TO 'utilizador'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON loja.ClienteContacto TO 'utilizador'@'localhost';  
GRANT SELECT ON loja.Jogo TO 'utilizador'@'localhost';  
GRANT SELECT ON loja.Jogo_Géneros TO 'utilizador'@'localhost';  
GRANT SELECT ON loja.Jogo_Desenvolvedores TO 'utilizador'@'localhost';  
GRANT SELECT (Num_Transac, Data_Transac, Qtd, Valor_Total, Tipo, ID_Cliente) ON loja.Transacção TO 'utilizador'@'localhost';  
GRANT SELECT (Num_Transac, Data_Transac, Qtd, Valor_Total, Tipo, ID_Cliente) ON loja.TransacçãoContem TO 'utilizador'@'localhost';
```

Figura 28 - Código SQL para a criação do perfil Utilizador

Para garantir a recuperação dos dados em caso de falha do sistema, é boa prática ter um sistema de *backup* regular. Procedemos à implementação deste sistema através da criação de um ficheiro *.bat*, o qual invoca o comando ***mysqldump*** passando como argumentos o nome da nossa *database*, o *host* e o *user* (e respetiva *password*) e guardando o ficheiro *.sql* de *backup* na diretoria definida. Não nos foi possível, ao invocar o comando *mysqldump*, escolher também a opção ‘*—routines*’ que permite guardar no ficheiro de *backup* os procedimentos e funções guardadas na nossa base de dados, devido a um erro que não conseguimos resolver, possivelmente graças à versão do servidor MySQL instalada não suportar essa opção no comando *mysqldump*. Ainda assim, em caso de necessidade de guardar os *stored procedures* e as *functions*, podemos utilizar uma alternativa de criação de ficheiros backup como, por exemplo, utilizar diretamente o *MySQL Workbench* para fazer “*Dump Database as SQL*”.

```
@echo off
setlocal

REM Definição das informações de acesso à base de dados MySQL
set MYSQL_HOST=localhost
set MYSQL_USER=clamideo
set MYSQL_PASSWORD=clamideo
set DATABASE=loja

REM Definição do caminho para a diretoria onde o backup será guardado
set BACKUP_DIR=C:\Users\BDBBackup

REM Definição do nome do ficheiro de backup
set BACKUP_FILENAME=%BACKUP_DIR%\backup_%date:~-4,4%%date:~-10,2%%date:~-7,2%_%time:~-11,2%%time:~-8,2%%time:~-5,2%.sql

REM Comando para executar o backup
mysqldump --host=%MYSQL_HOST% --user=%MYSQL_USER% --password=%MYSQL_PASSWORD% --databases %DATABASE% > %BACKUP_FILENAME%

REM Verificação do sucesso na execução do comando
if %errorlevel% neq 0 (
    echo Erro ao fazer o backup da base de dados.
    exit /b
)

echo Backup da base de dados realizado com sucesso.
exit /b
```

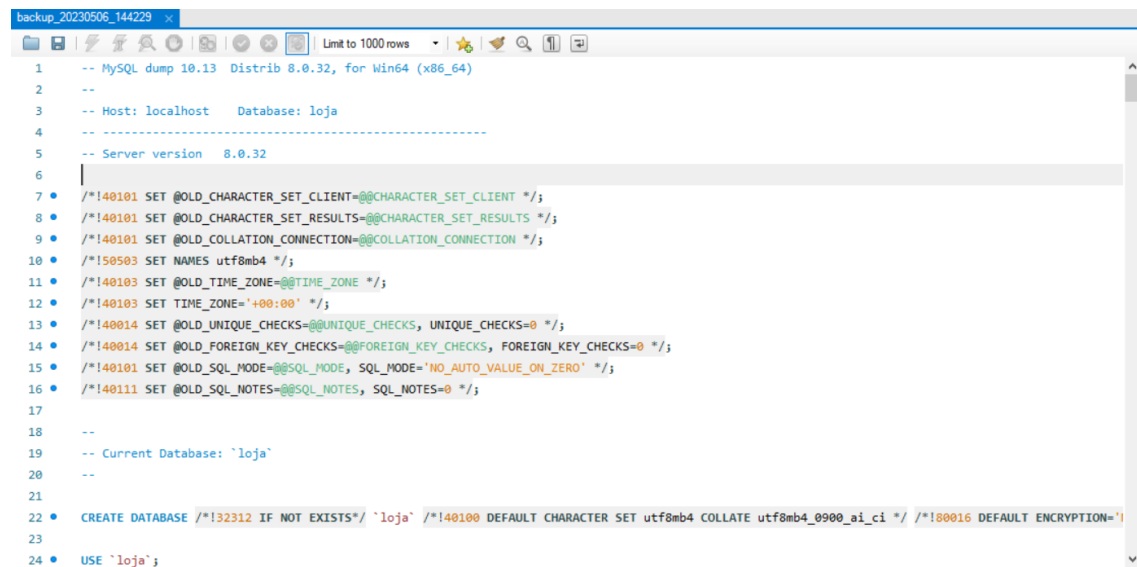
Figura 29 - Conteúdo do ficheiro backupScript.bat

Ao executar o ficheiro *.bat* na linha de comandos (Windows) obtemos uma mensagem de *Warning* (devido à possibilidade de ser inseguro passar a *password*) e a mensagem de sucesso da criação do *backup*.

```
>backupScript.bat
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Backup da base de dados realizado com sucesso.
```

Figura 30 - Execução do backupScript.bat

Localizando o ficheiro de backup *.sql* e abrindo-o no *MySQL Workbench* podemos verificar o backup efetuado com sucesso.



```
1  -- MySQL dump 10.13  Distrib 8.0.32, for Win64 (x86_64)
2  --
3  -- Host: localhost    Database: loja
4  --
5  -- Server version  8.0.32
6
7  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8  /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9  /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10 /*!50503 SET NAMES utf8mb4 */;
11 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12 /*!40103 SET TIME_ZONE='+00:00' */;
13 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17
18 --
19 -- Current Database: `loja`
20 --
21
22 CREATE DATABASE /*!32312 IF NOT EXISTS*/ `loja` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='A' */;
23
24 USE `loja`;
```

Figura 31 - Excerto inicial do ficheiro de backup *.sql*

6. Implementação do Sistema de Recolha de Dados

Estando o sistema criado e, satisfatoriamente, funcional, é altura de passar ao povoamento da nossa base de dados, onde vamos inserir dados nas tabelas, podendo utilizar os *stored procedures* para o efeito (bem como os *INSERT INTO* “manuais”). Com a base de dados povoada, poderemos também testar de um modo mais prático as *queries* de consulta criadas na secção 5.2, que respondem às interrogações do utilizador.

Para realizar o povoamento utilizámos, como vamos explicar com maior detalhe nas secções seguintes, dois métodos:

1. *Script SQL* que chama os *procedures* anteriormente criados e faz as inserções dos dados diretamente nas tabelas;
2. *Script Python* que se conecta ao *MySQL* e, importando um *dataset* escolhido e tratado previamente, transforma os dados lidos do *dataset* e insere-os na tabela Jogo da base de dados.

6.1. Apresentação e modelo do sistema

Como já foi referido, na implementação do nosso sistema de recolha de dados desenvolvemos dois métodos. O primeiro consistiu na criação de um *script SQL* que utiliza os *procedures* (e inserções) criados na secção 5.6 para povoar, numa fase inicial, as tabelas, inserindo 10 registos em cada uma das tabelas “Jogo”, “Cliente” e “Transação”. Para cada um destes procedimentos/inserções foram ainda inseridas entradas correspondentes nas tabelas “Jogo_Géneros”, “Jogo_Desenvolvedores”, “ClienteEmail”, “ClienteContacto” e “TransaçãoContem. Foram também inseridos dois registos na tabela Colaborador, correspondentes ao Sr. Clamídeo e ao Serginho.

A utilização de um segundo método deveu-se à necessidade de inserir uma maior quantidade de registos presentes num conjunto de dados guardado num ficheiro .csv na tabela Jogo (cerca de noventa milhares de entradas). Para isto seria extremamente ineficiente e desnecessariamente dispendioso em termos de tempo invocar no *script SQL* o procedimento *RegistarJogo* para cada um dos jogos, pelo que optamos por desenvolver um script em Python que permitisse fazer isto de forma automática, estabelecendo conexão ao *MySQL* e, após importar o *dataset*, invocar o procedimento guardado *RegistarJogo* para cada uma das linhas lidas do ficheiro. Assim, conseguimos em cerca de cinco minutos, fazer a inserção de, aproximadamente, noventa mil entradas na tabela Jogo.

6.2. Implementação do sistema de recolha

O primeiro passo foi, como já mencionado, o desenvolvimento de um script SQL que vai povoar inicialmente todas as tabelas através de invocações dos procedimentos implementados na secção 5.6:

```
CALL RegistrarCliente('Ana Silva', '1990-05-15', 'ana.silva@gmail.com', '123456789');
CALL RegistrarCliente('Pedro Santos', '1985-08-25', 'pedro.santos@gmail.com,pedro.santos2@gmail.com', '987654321');
CALL RegistrarCliente('Marta Pereira', '1992-11-10', 'marta.pereira@gmail.com', '456789123,987654321');
CALL RegistrarCliente('José Oliveira', '1982-03-18', 'jose.oliveira@gmail.com,jose.oliveira2@gmail.com', '321654987,654987321');
CALL RegistrarCliente('Sara Rodrigues', '1998-07-22', 'sara.rodrigues@gmail.com', '789123456');
CALL RegistrarCliente('André Costa', '1995-12-03', 'andre.costa@gmail.com', '654987321,123789456');
CALL RegistrarCliente('Mariana Almeida', '1991-09-08', 'mariana.almeida@gmail.com,mariana.almeida2@gmail.com', '123789456');
CALL RegistrarCliente('Carlos Santos', '1987-06-12', 'carlos.santos@gmail.com', '987321654');
CALL RegistrarCliente('Inês Silva', '1994-02-27', 'ines.silva@gmail.com', '456123789,789456123');
CALL RegistrarCliente('Rui Pereira', '1989-04-05', 'rui.pereira@gmail.com,rui.pereira2@gmail.com', '321987654');

CALL RegistrarJogo('The Witcher 3: Wild Hunt', 'Mature', 'PC', 'CD Projekt RED', 2015, 49.99, 10, 'RPG', 'CD Projekt RED');
CALL RegistrarJogo('Grand Theft Auto V', 'Mature', 'PS4', 'Rockstar Games', 2013, 59.99, 5, 'Action,Adventure', 'Rockstar North,Rockstar San Diego');
CALL RegistrarJogo('The Legend of Zelda: Breath of the Wild', '10+', 'Nintendo Switch', 'Nintendo', 2017, 59.99, 3, 'Action,Adventure', 'Nintendo');
CALL RegistrarJogo('Red Dead Redemption 2', 'Mature', 'Xbox One', 'Rockstar Games', 2018, 49.99, 8, 'Action,Adventure', 'Rockstar Games,Rockstar San Diego');
CALL RegistrarJogo('Minecraft', 'Everyone', 'PC', 'Mojang Studios', 2011, 29.99, 15, 'Sandbox', 'Mojang Studios');
CALL RegistrarJogo('Fortnite', 'Teen', 'Multiplataforma', 'Epic Games', 2017, 0, 20, 'Battle Royale', 'Epic Games,People Can Fly');
CALL RegistrarJogo('Super Mario Odyssey', 'Everyone', 'Nintendo Switch', 'Nintendo', 2017, 49.99, 5, 'Platform,Adventure', 'Nintendo');
CALL RegistrarJogo('The Last of Us Part II', 'Mature', 'PS4', 'Naughty Dog', 2020, 59.99, 2, 'Action,Adventure', 'Naughty Dog,Sony Interactive Entertainment');
CALL RegistrarJogo('League of Legends', 'Teen', 'PC', 'Riot Games', 2009, 0, 50, 'MOBA', 'Riot Games');
CALL RegistrarJogo('Call of Duty: Modern Warfare', 'Mature', 'Xbox One', 'Infinity Ward', 2019, 59.99, 4, 'First-person Shooter', 'Infinity Ward,Activision');
```

Figura 32 - Povoamento das tabelas Cliente, Jogo, ClienteEmail, ClienteContacto, Jogo_Géneros e Jogo_Desenvolvedores

```
INSERT INTO Colaborador (ID_Colab, Nome)
VALUES
(1, 'Sr. Clamídeo'),
(2, 'Serginho');
```

Figura 33 - Povoamento da tabela Colaborador

```
CALL RegistrarTransacao('2023-01-05', 1, 1, '1,1;3,1;', 'C');
CALL RegistrarTransacao('2023-02-10', 4, 1, '2,1;', 'C');
CALL RegistrarTransacao('2023-03-15', 5, 2, '4,2;1,1;', 'V');
CALL RegistrarTransacao('2023-01-25', 6, 1, '7,1;', 'C');
CALL RegistrarTransacao('2023-02-18', 2, 2, '5,1;1,1;', 'V');
CALL RegistrarTransacao('2023-03-05', 7, 2, '6,1;', 'C');
CALL RegistrarTransacao('2023-01-10', 3, 2, '2,2;1,2;', 'V');
CALL RegistrarTransacao('2023-02-28', 9, 2, '9,1;1,1;', 'C');
CALL RegistrarTransacao('2023-03-20', 10, 1, '8,1;', 'C');
CALL RegistrarTransacao('2023-01-15', 8, 2, '3,3;', 'V');
```

Figura 34 - Povoamento das tabelas Transação e TransaçãoContem

Já a implementação do *script* em *Python* foi iniciada com a tentativa de estabelecimento de conexão à base de dados, após proceder à instalação na máquina utilizada para o efeito do *MySQL connector* para *Python*. Após sucesso no estabelecimento da conexão, criamos um cursor para executar consultas na base de dados.

```
import mysql.connector

db = mysql.connector.connect(
    host="localhost",
    user="clamídeo",
    password="clamídeo",
    database="loja"
)

# Criar um cursor para executar consultas
cursor = db.cursor()
```

Figura 35 - Estabelecimento da conexão

Armazenamos o *path* para o ficheiro *.csv* na variável *csv_file* e abrimos o ficheiro, procedendo a ler linha a linha – ignorando a primeira linha (cabeçalho) – tendo em atenção o delimitador *‘;’*. Para cada linha, armazenamos em variáveis os dados correspondentes a cada coluna (separados pelo delimitador) e, no final, invocamos, através do cursor, o procedimento *RegistarJogo*, passando as variáveis criadas. Fazemos ainda *commit* para guardar as alterações.

```
# Abrir o ficheiro CSV
with open(csv_file, 'r', encoding="utf-8") as file:
    # Ler o ficheiro CSV
    csv_reader = csv.reader(file, delimiter=';')

    # Saltar o cabeçalho do ficheiro
    next(csv_reader)

    # Iterar sobre as linhas do ficheiro
    for row in csv_reader:
        # Extrair os dados de cada coluna
        nome = str(row[0])[:50] # Limitar a 50 caracteres
        generos = row[1]
        esrb = str(row[2])[:8] # Limitar a 8 caracteres
        plataforma = row[3]
        publicadora = str(row[4])[:30] # Limitar a 30 caracteres
        desenvolvedores = row[5].replace(" / ", ",") # Substituir " / " por ","
        ano_lanc = int(row[6]) if row[6].isdigit() else 0 # Atribuir 0 se o valor for inválido ou vazio
        preco = float(row[7])
        qtd = int(row[8])

        # Invocar o procedimento RegistarJogo passando os dados do jogo
        cursor.callproc('RegistarJogo', [nome, esrb, plataforma, publicadora, ano_lanc, preco, qtd, generos, desenvolvedores])

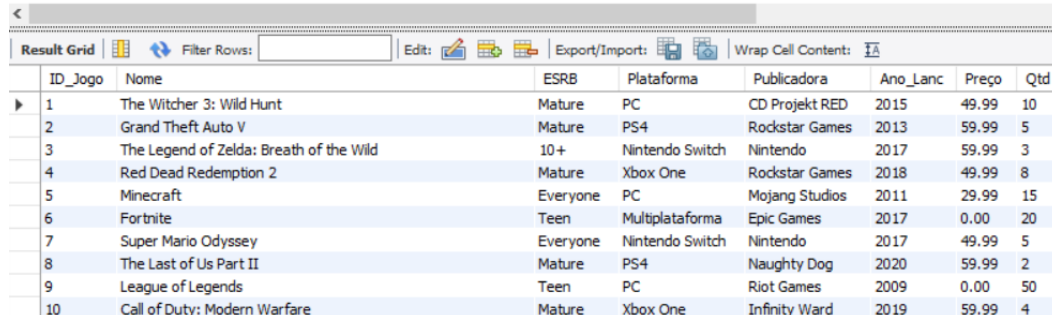
    # Commit para guardar as alterações
    db.commit()
```

Figura 36 - código python para povoamento da tabela Jogo

6.3. Funcionamento do sistema

Após execução de ambos os *scripts* criados, procedemos a verificar se os dados realmente são inseridos como esperado nas tabelas *Jogo*, *Jogo_Gêneros* e *Jogo_Desenvolvedores*, obtendo uma confirmação positiva, verificando que em todas as três tabelas foram bem sucedidas as invocações do procedimento *RegistrarJogo* em ambos os *scripts SQL* e *Python*. Na figura 37, podemos observar os dez primeiros registos da tabela *Jogo*, que foram todos eles inseridos pelo *script SQL*.

```
28 • SELECT *
29 FROM Jogo
30 LIMIT 100000;
```

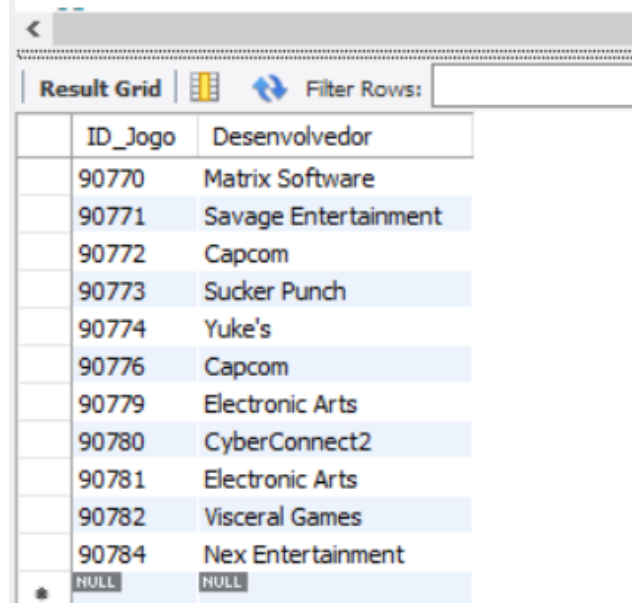


ID_Jogo	Nome	ESRB	Plataforma	Publicadora	Ano_Lanc	Preço	Qtd
1	The Witcher 3: Wild Hunt	Mature	PC	CD Projekt RED	2015	49.99	10
2	Grand Theft Auto V	Mature	PS4	Rockstar Games	2013	59.99	5
3	The Legend of Zelda: Breath of the Wild	10+	Nintendo Switch	Nintendo	2017	59.99	3
4	Red Dead Redemption 2	Mature	Xbox One	Rockstar Games	2018	49.99	8
5	Minecraft	Everyone	PC	Mojang Studios	2011	29.99	15
6	Fortnite	Teen	Multiplataforma	Epic Games	2017	0.00	20
7	Super Mario Odyssey	Everyone	Nintendo Switch	Nintendo	2017	49.99	5
8	The Last of Us Part II	Mature	PS4	Naughty Dog	2020	59.99	2
9	League of Legends	Teen	PC	Riot Games	2009	0.00	50
10	Call of Duty: Modern Warfare	Mature	Xbox One	Infinity Ward	2019	59.99	4

Figura 37 - Conteúdo inserido na tabela *Jogo*

Já na figura 38 podemos observar os últimos registos da tabela *Jogo_Desenvolvedores*, que correspondem às últimas entradas do *dataset* com desenvolvedores associados.

```
32 • SELECT *
33 FROM Jogo_Desenvolvedores
34 LIMIT 100000;
```



ID_Jogo	Desenvolvedor
90770	Matrix Software
90771	Savage Entertainment
90772	Capcom
90773	Sucker Punch
90774	Yuke's
90776	Capcom
90779	Electronic Arts
90780	CyberConnect2
90781	Electronic Arts
90782	Visceral Games
90784	Nex Entertainment
NULL	NULL

Figura 38 - Últimos registos da tabela *Jogo_Desenvolvedores*

As restantes tabelas também foram povoadas com sucesso, a partir do *script SQL*, como exemplifica a figura 39.

```

59 • SELECT tc.Num_Transac, tc.ID_Jogo, tc.Qtd, tc.Valor_Total, t.Qtd AS Soma_Qtds, t.Valor_Total AS Soma_Valores_Totais
60 FROM TransaçãoContem tc
61 JOIN Transação t ON tc.Num_Transac = t.Num_Transac;

```

	Num_Transac	ID_Jogo	Qtd	Valor_Total	Soma_Qtds	Soma_Valores_Totais
1	1	1	1	49.99	2	109.98
1	3	1	1	59.99	2	109.98
2	2	1	1	59.99	1	59.99
3	1	1	1	49.99	3	149.97
3	4	2	2	99.98	3	149.97
4	7	1	1	49.99	1	49.99
5	1	1	1	49.99	2	79.98
5	5	1	1	29.99	2	79.98
6	6	1	1	0.00	1	0.00
7	1	2	2	99.98	4	219.96
7	2	2	2	119.98	4	219.96
8	1	1	1	49.99	2	49.99
8	9	1	1	0.00	2	49.99
9	8	1	1	59.99	1	59.99
10	3	3	3	179.97	3	179.97

Figura 39 - Conteúdo das tabelas Transação e TransaçãoContem

7. Conclusões e Trabalho Futuro

Com a realização deste trabalho conseguimos atingir a grande maioria dos objetivos propostos, apesar de não termos produzido a última fase, correspondente à implementação do sistema de painéis de análise, que pode, no futuro, ser executada.

Consideramos ter realizado um trabalho bastante satisfatório, bem planeado e estruturado, principalmente na definição do sistema – na qual conseguimos relevar todos os pontos importantes na contextualização, fundamentação, motivação, análise de viabilidade e exposição dos recursos necessários, equipa de trabalho e plano de execução – na modelação de diagramas que nos permitiram iniciar a visualização daquilo que viria a ser o nosso produto final e na implementação física do nosso Sistema de Bases de Dados, cujo resultado excedeu, nas nossas humildes opiniões, as expectativas iniciais.

Apesar disso, houve também aspetos negativos, dos quais se destaca a – já referida – não implementação do sistema de painéis de análise, que achamos ser capazes de desenvolver, o que não foi possível – e este foi outro dos pontos menos positivos do desenvolvimento do projeto – graças a uma organização do tempo razoavelmente abaixo do pretendido. Também existem certos excertos do relatório que poderiam ser melhorados, principalmente nas explicações das metodologias e processos seguidos.

Não obstante e em suma, achamos ter demonstrado boas capacidades para desenvolver um projeto de dimensão média-longa e boas capacidades de inovação, adaptação e resolução de problemas através da pesquisa, exploração e aprendizagem de novos conhecimentos.

Referências

Connolly, T., Begg, C., Database Systems: A Practical Approach to Design, Implementation, and Management, AddisonWesley, Global Edition, 26 Sep 2014. ISBN-10: 1292061189, ISBN-13: 978-1292061184.

Belo, O., “Bases de Dados Relacionais: Implementação com MySQL”, FCA – Editora de Informática, 376p, Set 2021. ISBN: 978-972-722-921-5