



Instituto Politécnico do Cávado e do Ave
Escola Superior de Tecnologia
Licenciatura em Engenharia de Sistemas Informáticos

Trabalho Prático I: Processos de ETL
Disciplina: Integração de Sistemas de Informação
Ano letivo 2024/2025

Victor Destefani - 18586

Docente: Luís Ferreira

Barcelos

Outubro/2024

Índice

Índice de Figuras	3
1. Introdução	4
2. Contextualização	5
3. Estratégia Utilizada CSV to XML	6
3.1. Entrada de Arquivo CSV	6
3.2. Incremento de ID.....	6
3.3. Seleção e Filtragem de Valores	6
3.4. Validação dos Dados	6
3.5. Criptografia dos Dados	6
3.6. Descriptografia dos Dados	6
3.7. Notificação por Email	7
4. Transformações.....	8
4.1 Diagramas de Transformação CSV to XML	8
4.2. Diagramas de Transformação CSV to MySql	9
5. Jobs.....	11
5.1 Unificação dos Processos ETL.....	11
6. Integração Web API e ASP NET Core MVC	13
7. Vídeo com demonstração (QR Code)	17
8. Conclusão	18
9. Bibliografia	19

Índice de Figuras

Figura 1 - Configuração SMTP para alertas.	7
Figura 2 – Fluxo do Processo de Transformação CSV para XML.	8
Figura 3 - Fluxo do Processo de Transformação CSV para MySQL.	9
Figura 4 - Exemplo de criação de tabela no MySql pelo Pentaho.	10
Figura 5 - Job com a Unificação dos Processos ETL.....	11
Figura 6 - Receção do e-mail com anexo do XML.	12
Figura 7 - Classe abstrata implementada na API.....	13
Figura 8 - Classe DTO na API.	14
Figura 9 - Função genérica na API para busca de resultados de vendas.....	14
Figura 10 - Classe MyDbContext e seu Mapeamento com o Banco de Dados.	15
Figura 11 - Controlador de vendas, para comunicação com a API.....	15
Figura 12 - Classe ApiVendasService que consome os dados da API.	16
Figura 13 - Busca e resultado em interface web.....	16
Figura 14 - QR Code que encaminha para vídeo.....	17

1. Introdução

Na era da informação, a integração de sistemas é fundamental para que uma organização possa se destacar e manter sua competitividade. A habilidade de reunir e unificar dados provenientes de diversas fontes, convertê-los em informações úteis e automatizar tarefas tornou-se uma prioridade para empresas de todos os setores. A disciplina de Integração de Sistemas de Informação (ISI), parte do curso de Engenharia de Sistemas de Informáticos, tem como objetivo preparar e capacitar os alunos para entender e aplicar os principais conceitos e ferramentas necessários à integração de dados e processos dentro das organizações.

Este relatório descreve um projeto desenvolvido no âmbito da Unidade Curricular de Integração de Sistemas de Informação (ISI). Através da utilização do software *Pentaho Data Integration* (PDI), o projeto foca-se na integração de sistemas, um processo essencial para conectar e harmonizar dados e processos provenientes de diferentes fontes. No ambiente empresarial, a integração de sistemas desempenha um papel crucial, promovendo a otimização da tomada de decisões, a melhoria da eficiência operacional e o estímulo à inovação.

O projeto foi iniciado a partir de uma solicitação do Diretor da empresa XYZ, que identificou a necessidade de aprimorar a gestão das informações sobre jogos de diversas consolas lançados entre os anos 2000 e 2015. A empresa XYZ buscava criar uma base de dados em português, filtrando apenas as informações essenciais para alimentar essa nova base, destinada a uma de suas filiais em Portugal. Para atender a essa demanda, o *Pentaho Data Integration* (PDI) foi escolhido como uma solução versátil para integração de dados. Com o uso do PDI, foi possível implementar uma solução eficiente, facilitando a extração, transformação e carregamento (ETL) dos dados, além de automatizar processos críticos para a empresa.

Ao longo deste relatório, exploraremos como o *Pentaho Data Integration* (PDI) desempenhou um papel central na solução do desafio apresentado pelo Diretor da empresa XYZ, destacando a relevância da integração de sistemas de informação no cenário empresarial atual. Além disso, abordaremos a estratégia adotada, os operadores e processos envolvidos, bem como a criação de uma API personalizada para tratar e gerenciar os novos dados na base de dados da filial em Portugal. Discutiremos também os principais aprendizados obtidos durante a resolução bem-sucedida deste problema, oferecendo uma visão detalhada de como o PDI, em conjunto com a API, se tornou essencial para a integração eficaz de sistemas de informação na empresa.

2. Contextualização

O ambiente de negócios em constante transformação e a revolução da informação colocam as empresas frente a desafios intrincados. A habilidade de aceder, tratar e disseminar informações de forma eficiente tornou-se um elemento crucial para o êxito das organizações. O desafio discutido neste projeto espelha uma circunstância frequente que organizações enfrentam ao tentar aprimorar a administração de dados e a automação de procedimentos em um cenário de alta competitividade.

O Diretor da empresa XYZ expressou a necessidade de criar uma base de dados eficiente, contendo informações detalhadas sobre jogos lançados entre os anos 2000 e 2015. Para isso, a sede da empresa forneceu um ficheiro .csv com os dados a serem tratados e integrados na filial em Portugal. O desafio exigiu a utilização de um processo ETL (Extração, Transformação e Carregamento) que incluiu a filtragem das colunas necessárias, a criação de um arquivo XML com os dados filtrados e a validação deste arquivo por meio de um XSD, garantindo a conformidade e integridade dos dados.

Além disso, os dados resultantes desse processo foram carregados diretamente na base de dados da filial a partir do ficheiro .csv, assegurando que a integração fosse realizada de forma automatizada e eficiente. Antes do envio dos dados para a base de dados, foi realizada uma checagem inicial dos dados brutos para identificar possíveis inconsistências. Em seguida, os dados foram validados conforme a estrutura das tabelas e os tipos de variáveis definidos no sistema. Somente após essa validação, os dados filtrados e ajustados foram inseridos diretamente na base de dados MySQL, garantindo a consistência e qualidade das informações armazenadas.

Neste documento, vamos investigar a tática empregada para lidar com esse desafio, os operadores e processos implicados, bem como as principais lições obtidas com a implementação bem-sucedida desta solução no âmbito do *Pentaho* e da Unidade Curricular ISI.

3. Estratégia Utilizada CSV to XML

A estratégia adotada para abordar o problema apresentado pelo administrador da empresa XYZ envolveu uma abordagem estruturada, com validação de dados e automação de processos. A solução foi implementada com o uso do software *Pentaho Data Integration* (PDI), utilizando uma série de operadores e processos bem definidos. Abaixo está um resumo do fluxo:

3.1. Entrada de Arquivo CSV

O primeiro passo foi o uso do componente "CSV file input", que permitiu a leitura do ficheiro .csv fornecido pela sede da empresa. Este ficheiro continha dados de jogos lançados entre 2000 e 2015. No componente, foi especificado o delimitador por "virgula", o "enclosure" (para campos com aspas), e foram configuradas as colunas para garantir que os tipos de dados fossem corretamente interpretados, como String, Integer e Boolean.

3.2. Incremento de ID

O processo seguinte foi a aplicação do operador "IncrementId", que gerou identificadores únicos para cada linha de dados, permitindo uma melhor organização e rastreamento dos registos inseridos no sistema.

3.3. Seleção e Filtragem de Valores

Em seguida, o operador "Select values" foi utilizado para filtrar apenas as colunas necessárias. A empresa precisava apenas de informações essenciais para sua base de dados em Portugal, como o título dos jogos, número máximo de jogadores, gêneros e se eram compatíveis com múltiplas plataformas. Assim, colunas irrelevantes foram removidas.

3.4. Validação dos Dados

Primeiramente, o job valida os dados convertendo o CSV em XML e verificando se o arquivo segue o formato correto, validando também a conformidade com um esquema XSD. Paralelamente, é feita a verificação da conexão com o banco de dados, assegurando que as tabelas e colunas estejam configuradas corretamente.

3.5. Criptografia dos Dados

Utiliza criptografia simétrica (AES) para criptografar dados específicos. Esta etapa é crucial para salvaguardar as informações, garantindo que elas permaneçam ilegíveis sem a chave de descryptografia adequada. A chave de criptografia é protegida e só será utilizada novamente durante o processo de descryptografia.

3.6. Descryptografia dos Dados

Para garantir a segurança e reversibilidade da criptografia, o componente "descryptografar dados" realiza o processo de descryptografia dos dados criptografados. Neste cenário, é utilizada a mesma chave utilizada na criptografia, garantindo que os dados sejam restaurados ao seu estado original.

3.7. Notificação por Email

Com o intuito de gerar um alerta sobre o processo de conversão do CSV para XML, foi implementado o envio desse alerta por e-mail. Para isso, foi necessário configurar o servidor SMTP (Simple Mail Transfer Protocol), que é responsável pelo envio de mensagens de e-mail. Essa configuração permitiu que os alertas fossem encaminhados automaticamente para os destinatários especificados, notificando-os sobre o resultado do processo de conversão.

A configuração do servidor SMTP envolveu a definição de parâmetros essenciais, como:

- O endereço do servidor SMTP (host).
- A porta de comunicação utilizada.
- As credenciais de autenticação (usuário e senha).
- E outros detalhes de segurança, como o uso de SSL/TLS para garantir a integridade e a proteção dos dados durante a transmissão.

Dessa forma, sempre que o processo de conversão do CSV para XML fosse executado, o alerta seria disparado por e-mail juntamente com o XML gerado, permitindo que os responsáveis acompanhassem o status em tempo real e fossem notificados imediatamente em caso de sucesso ou falha no processo.

The screenshot shows a 'Mail' configuration window with a title bar containing a mail icon and the text 'Mail'. Below the title bar, there is a tabbed interface with four tabs: 'Addresses', 'Server', 'EMail Message', and 'Attached Files'. The 'Server' tab is currently selected. Under the 'Server' tab, there are two main sections: 'SMTP Server' and 'Authentication'. In the 'SMTP Server' section, the 'SMTP Server' field is set to 'smtp.office365.com' and the 'Port' field is set to '25'. In the 'Authentication' section, the 'Use authentication?' checkbox is checked, the 'Authentication user' field is set to 'a18586@alunos.ipca.pt', the 'Authentication password' field is masked with dots, the 'Use secure authentication?' checkbox is checked, and the 'Secure connection type' dropdown menu is set to 'TLS'.

Figura 1 - Configuração SMTP para alertas.

4. Transformações

4.1 Diagramas de Transformação CSV to XML

Pode-se observar melhor o esquema de como está a ser implementado a transformação do CSV para o XML, na qual os passos estão a ser todos validados em cada sessão.

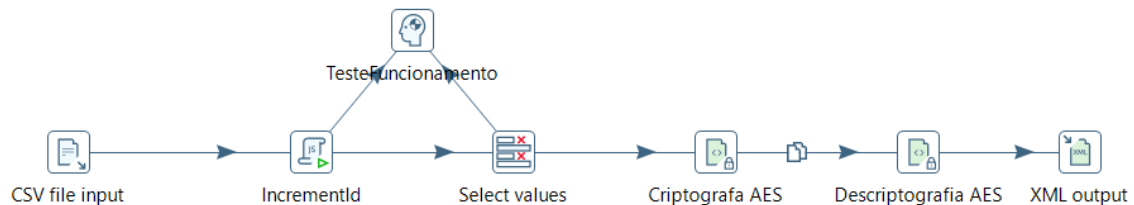


Figura 2 – Fluxo do Processo de Transformação CSV para XML.

CSV File Input (Entrada de Arquivo CSV):

- Este passo indica a entrada de um arquivo CSV, que contém dados brutos que serão processados.
- O componente de entrada de CSV lê o arquivo e os dados contidos nele.

IncrementId (Incrementar ID):

- Este passo incrementa um ID aos dados do CSV, usado para criar um campo exclusivo para rastrear registos de maneira única.
- É um passo de transformação onde um novo campo ID é adicionado ao conjunto de dados.

Select Values (Selecionar Valores):

- Seleção de determinados campos do conjunto de dados. Este passo permite eliminar ou manter apenas as colunas que são necessárias para os passos seguintes.
- É um passo fundamental para limpeza de dados ou preparação para encriptação, limitando as informações que serão processadas.

TesteFuncionamento:

- Componente de validação e verificação do funcionamento do fluxo.
- É utilizado para verificar se os dados estão corretos até este ponto antes de prosseguir para a próxima fase.

Criptografa AES (Criptografia AES):

- Neste passo, os dados são criptografados usando o algoritmo AES (Advanced Encryption Standard), um algoritmo de criptografia simétrica.
- A criptografia AES é utilizada para garantir a segurança dos dados antes de prosseguir para o próximo passo.

Descriptografia AES:

- Após a criptografia, os dados são descriptografados usando o mesmo algoritmo AES.
- Este passo restaura os dados ao seu estado original, garantindo que o processo de criptografia e descriptografia foi bem-sucedido.

XML Output (Saída XML):

- Finalmente, o resultado do processamento é exportado em formato XML.
- Este passo gera um arquivo XML como saída, provavelmente com os dados que passaram por transformação e encriptação.

Esse fluxo demonstra um processo típico onde dados são importados, transformados, criptografados, validados e, por fim, exportados em um formato estruturado (XML).

4.2. Diagramas de Transformação CSV to MySql

O fluxo começa com a entrada de dados brutos a partir de um arquivo CSV. Esses dados passam por um processo de transformação, onde um campo incremental é adicionado para identificar cada registo de forma única. A seguir, são selecionados os campos ou colunas relevantes para o processamento. Em uma etapa posterior, esses dados são validados contra uma tabela de referência em um banco de dados MySQL, garantindo que estejam em conformidade com a estrutura esperada.

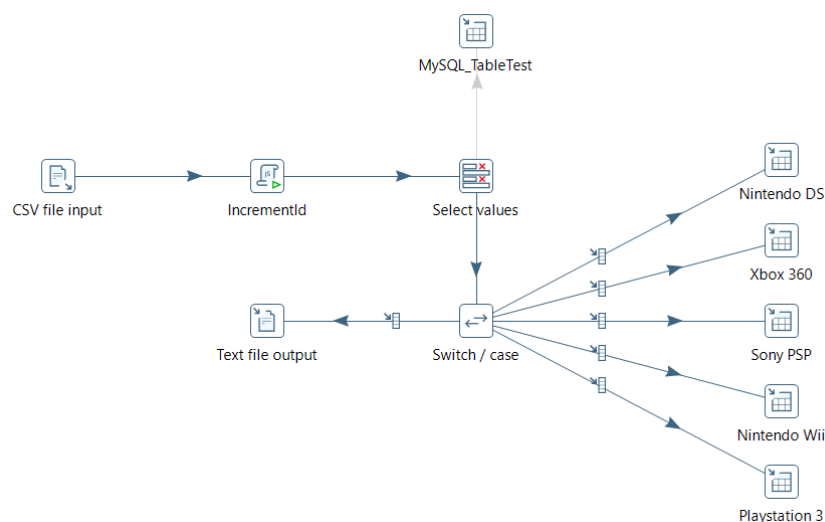


Figura 3 - Fluxo do Processo de Transformação CSV para MySQL.

NOTA: Os pontos **CSV file input**, **IncrementId** e **Select Values**, funcionam de igual modo ao modelo anterior de transformação (CSV to XML).

MySQL_TableTest (Validação de Colunas):

- Os dados selecionados são validados em relação a uma tabela de referência genérica em um banco de dados MySQL. O objetivo desta etapa é verificar se os dados têm as colunas corretas e a estrutura esperada antes de serem processados ou roteados para diferentes saídas. Garantindo que os dados estejam em conformidade com o modelo de dados esperado.

Switch / Case (Roteamento Condicional):

- O "Switch / Case" é um componente condicional que determina o destino dos dados com base em uma regra específica. A regra baseia-se em um campo do conjunto de dados (tipo de console), que determina para qual saída o registro será direcionado. Dependendo do valor desse campo, o fluxo segue para uma das várias saídas disponíveis, cada uma representando um console específico.

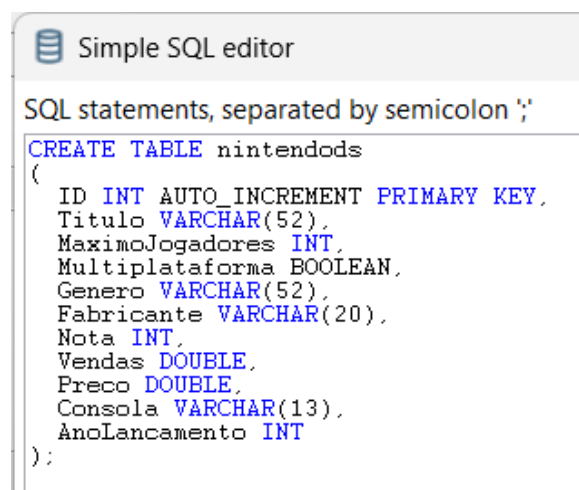
Nintendo DS, Xbox 360, Sony PSP, Nintendo Wii, Playstation 3 (Saídas Específicas):

- Cada uma dessas saídas corresponde a um destino onde os dados são armazenados ou processados com base na regra aplicada no "Switch / Case". Por exemplo, registros relacionados ao Nintendo DS são roteados para a saída correspondente ao Nintendo DS, enquanto os dados de outros consoles seguem para suas respectivas saídas.

Text File Output (Saída de Arquivo de Texto):

- Para os registros que não atendem a nenhuma das condições definidas no "Switch / Case", há uma saída alternativa. Esses registros são salvos em um arquivo de texto, que serve como um repositório para os dados que não se enquadram nas regras predefinidas.

Em resumo, este fluxo descreve um processo de transformação e roteamento de dados, onde os dados são validados, filtrados e, com base em critérios específicos, direcionados para diferentes destinos. Os registros que não atendem aos critérios são armazenados separadamente para posterior análise ou processamento. Para a criação das tabelas, o próprio Pentaho proporciona essa opção, de tal modo que podemos ter uma otimização no processo.



```
CREATE TABLE nintendods
(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Titulo VARCHAR(52),
  MaximoJogadores INT,
  Multiplataforma BOOLEAN,
  Genero VARCHAR(52),
  Fabricante VARCHAR(20),
  Nota INT,
  Vendas DOUBLE,
  Preco DOUBLE,
  Consola VARCHAR(13),
  AnoLancamento INT
);
```

Figura 4 - Exemplo de criação de tabela no MySql pelo Pentaho.

5. Jobs

5.1 Unificação dos Processos ETL

A figura 5 apresentada, descreve um **JOB** no contexto de um fluxo de trabalho de ETL, representando um processo automatizado para transformar e validar dados, além de garantir que tudo esteja correto antes de exportar os resultados para diferentes saídas. Esse **JOB** foca na conversão de arquivos CSV para XML e MySQL, além de realizar validações com base em XSD (esquema XML) e verificações no banco de dados. Dependendo do sucesso ou falha dessas validações, o fluxo segue diferentes caminhos, podendo ser interrompido ou finalizado com sucesso.

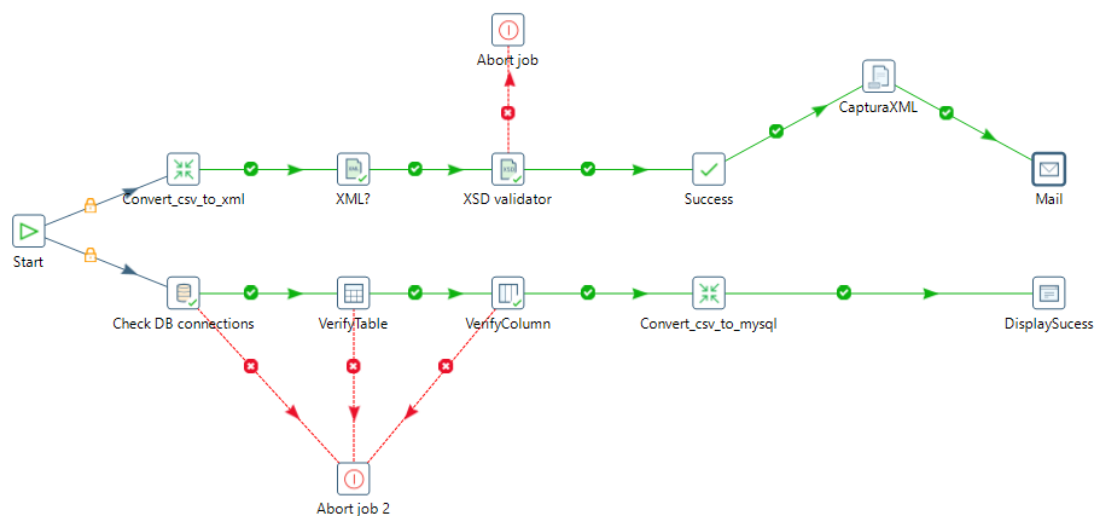


Figura 5 - Job com a Unificação dos Processos ETL.

Start:

- O ponto de início do **JOB**, onde o processo de automação é disparado. A partir daqui os diferentes caminhos começam a se ramificar.

Convert_csv_to_xml:

- Este passo converte os dados de um arquivo CSV para o formato XML. O arquivo CSV original é transformado em um XML, que pode ser usado nas etapas subsequentes.

XML? (Decisão se o arquivo é XML):

- Um ponto de decisão para verificar se o arquivo convertido para XML está no formato correto. Se for um XML válido, o fluxo prossegue para o próximo passo, caso contrário, pode interromper o **JOB**.

XSD Validator (Validação XSD):

- Aqui, o arquivo XML gerado é validado contra um esquema XSD. Esta etapa garante que a estrutura e o formato do XML estejam de acordo com um padrão predefinido (XSD). Se houver alguma inconsistência, o **JOB** pode ser interrompido.

Abort Job (Aborto do JOB em caso de falha de validação):

- Se o arquivo XML não passar pela validação XSD, o **JOB** é interrompido, indicando uma falha no processo e evitando que os passos seguintes sejam executados. Isso protege a integridade dos dados.

Success (Sucesso na validação):

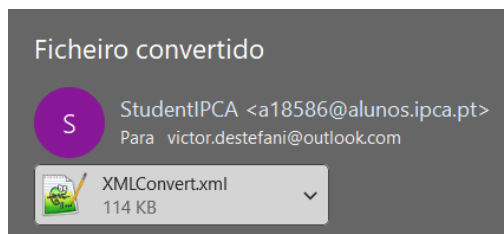
- Se a validação do XSD for bem-sucedida, o fluxo segue para a conclusão, indicando que o arquivo XML é válido e o processo pode continuar.

CapturaXML (Captura de XML):

- Esta etapa captura e armazena o XML validado para ser usado em passos futuros ou como resultado. Esse XML pode ser necessário para análises, armazenamento ou envio.

Mail (Envio de E-mail):

- Uma vez que o arquivo XML tenha sido validado e capturado com sucesso, uma notificação é enviada por e-mail, informando o sucesso do processo.



Bom dia, ficheiro convertido.

Job:

JobName : myJob

Directory : /

JobEntry : Mail

Figura 6 - Receção do e-mail com anexo do XML.

Check DB Connections (Verificação de Conexões de Banco de Dados):

- Em paralelo à conversão para XML, o **JOB** verifica as conexões com o banco de dados. Esta etapa garante que a comunicação com o banco de dados está funcionando corretamente antes de realizar operações adicionais.

VerifyTable (Verificação da Tabela):

- Após a verificação de conexão, o **JOB** verifica a existência da tabela no banco de dados, garantindo que a estrutura de dados onde as informações serão inseridas está presente.

VerifyColumn (Verificação de Colunas):

- Esta etapa verifica se as colunas específicas na tabela estão presentes e configuradas corretamente, evitando problemas ao inserir dados no banco.

Abort Job 2 (Aborto do JOB em caso de falha no banco de dados):

- Se houver qualquer problema nas conexões de banco de dados, nas tabelas ou nas colunas, o **JOB** é interrompido, garantindo que nenhum dado seja inserido em um ambiente não configurado corretamente.

Convert_csv_to_mysql (Conversão do CSV para MySQL):

- Se as verificações no banco de dados forem bem-sucedidas, o arquivo CSV é convertido para um formato que pode ser inserido no MySQL. Aqui, os dados do CSV são preparados e transformados para se adequarem à estrutura do banco de dados MySQL.

DisplaySuccess (Exibir Sucesso):

- Após a conclusão bem-sucedida de todas as etapas, o **JOB** exibe uma mensagem de sucesso, confirmando que os dados foram convertidos e validados corretamente.

6. Integração Web API e ASP NET Core MVC

Para a manipulação dos dados criados durante o processo do ETL, foi implementada duas soluções utilizando a linguagem C#, sendo elas a criação de uma API Web do ASP.NET Core e um Aplicativo Web do ASP .NET Core (MVC). Ambas seguem uma estrutura cuidadosamente organizada que visa expor e apresentar os dados de vendas de consoles de videogame. Todo o processo está relacionado com uma fase anterior de **ETL (Extração, Transformação e Carga)**, garantindo que os dados sejam validados e estruturados adequadamente antes de serem apresentados ao utilizador final.

Como exemplo de demonstração, foi implementada na API uma classe abstrata *JogosAbstract*, que define a estrutura básica de um jogo, que inclui atributos como **ID**, **Título**, **Gênero**, **Fabricante**, entre outros. Isso assegura que as informações dos jogos estejam organizadas de maneira consistente no banco de dados.

```
public abstract class JogosAbstract
{
    [Key] // Indica que esta propriedade é a chave primária
    0 referências
    public int ID { get; set; } //OK

    0 referências
    public string? Titulo { get; set; } //OK
    1 referência
    public double Vendas { get; set; } // OK
    0 referências
    public double Preco { get; set; } // OK
    0 referências
    public int Nota { get; set; } // OK
    0 referências
    public bool Multiplataforma { get; set; } // OK
    0 referências
    public int MaximoJogadores { get; set; } // OK
    0 referências
    public string? Genero { get; set; } // OK
    0 referências
    public string? Fabricante { get; set; } //OK
    1 referência
    public string? Consola { get; set; } //OK
    2 referências
    public int AnoLancamento { get; set; } // OK
}
```

Figura 7 - Classe abstrata implementada na API.

A API utiliza um **DTO (Data Transfer Object)** chamado `ConsolaVendaDTO`, que abstrai os dados de vendas de cada consola, contendo propriedades como **TotalVendas** e **Consola**. Essa abordagem ajuda a formatar e transmitir dados entre o cliente e o servidor de forma eficiente.

```
namespace MyAPI.ViewModel
{
    public class ConsolaVendaDTO
    {
        public string Consola { get; set; }

        public double TotalVendas { get; set; }

        public string TotalVendasFormatted { get; set; }
    }
}
```

Figura 8 - Classe DTO na API.

A API oferece pontos de extremidade (endpoints) específicos para consultar as vendas de cada tipo de consola, como Nintendo DS, Playstation 3, Sony PSP, entre outros. Cada um desses endpoints é projetado para aceitar parâmetros de período, permitindo que o utilizador consulte as vendas em um intervalo de tempo determinado.

No código do `ConsolaVendasController`, há métodos genéricos e específicos para consoles como `GetVendasNintendoDS`, `GetVendasPlaystation3`, etc., que utilizam o método `ObterVendasPorConsola` para realizar a consulta das vendas.

```
// Esta função lida com vendas específicas de consolas
5 referências
public async Task<ActionResult<List<ConsolaVendaDTO>>> ObterVendasPorConsola<T>(DateTime inicio, DateTime fim) where T : JogosAbstract
{
    var vendas = await _context.Set<T>()
        .Where(j => j.AnoLancamento >= inicio.Year && j.AnoLancamento <= fim.Year)
        .GroupBy(j => j.Consola)
        .Select(grupo => new ConsolaVendaDTO
        {
            Consola = grupo.Key,
            TotalVendas = grupo.Sum(j => j.Vendas)
        }).ToListAsync();

    // Formatar TotalVendas para duas casas decimais
    foreach (var venda in vendas)
    {
        venda.TotalVendasFormatted = venda.TotalVendas.ToString("F2", CultureInfo.InvariantCulture);
    }

    return Ok(vendas);
}

// Métodos específicos para Nintendo DS
[HttpGet("nintendods/periodo")]
0 referências
public async Task<ActionResult<List<ConsolaVendaDTO>>> GetVendasNintendoDS(DateTime inicio, DateTime fim)
{
    return await ObterVendasPorConsola<NintendoDS>(inicio, fim);
}

// Métodos específicos para Nintendo Wii
[HttpGet("nintendowii/periodo")]
0 referências
public async Task<ActionResult<List<ConsolaVendaDTO>>> GetVendasNintendoWii(DateTime inicio, DateTime fim)
{
    return await ObterVendasPorConsola<NintendoWii>(inicio, fim);
}
```

Figura 9 - Função genérica na API para busca de resultados de vendas.

O **MyDbContext** foi configurado para mapear entidades específicas de cada console (NintendoDS, Playstation3, SonyPSP, etc.) em tabelas distintas do banco de dados, seguindo a estratégia de herança Table Per Type (TPT). Isso permite que cada console tenha sua própria tabela, mas todas as tabelas compartilhem uma estrutura comum derivada da classe JogosAbstract.

```
7 referências
public class MyDbContext : Microsoft.EntityFrameworkCore.DbContext
{
    0 referências
    public MyDbContext(DbContextOptions<MyDbContext> options) : base(options) { }

    //public DbSet<JogosAbstract> Jogos { get; set; }
    0 referências
    public DbSet<NintendoDS> DsJogos { get; set; }
    0 referências
    public DbSet<NintendoWii> WiiJogos { get; set; }
    0 referências
    public DbSet<Playstation3> Play3Jogos { get; set; }
    0 referências
    public DbSet<SonyPSP> PspJogos { get; set; }
    0 referências
    public DbSet<Xbox360> XboxJogos { get; set; }

    //Mapear cada subclasse para sua própria tabela (estratégia Table Per Type - TPT)

    0 referências
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        // Define herança usando TPT (Table per Type)
        modelBuilder.Entity<NintendoDS>().ToTable("nintendods");
        modelBuilder.Entity<NintendoWii>().ToTable("nintendowii");
        modelBuilder.Entity<Playstation3>().ToTable("playstation3");
        modelBuilder.Entity<SonyPSP>().ToTable("sonypsp");
        modelBuilder.Entity<Xbox360>().ToTable("xbox360");
    }
}
```

Figura 10 - Classe MyDbContext e seu Mapeamento com o Banco de Dados.

A página web foi criada para permitir que os utilizadores finais acessem os dados de vendas diretamente no navegador. O controlador VendasController se comunica com a API para buscar os dados de vendas com base nas datas inseridas pelo utilizador. O controlador VendasController valida as entradas de data, garantindo que o formato correto seja utilizado e que a API retorne os dados adequados. Caso os dados não sejam encontrados ou estejam incorretos, mensagens de erro apropriadas são exibidas na interface.

```
[HttpPost]
0 referências
public async Task<IActionResult> Index(string consola, string inicio, string fim)
{
    DateTime dataInicio, dataFim;

    // Validação das datas recebidas no formato "dd/MM/yyyy"
    if (!DateTime.TryParseExact(inicio, "dd/MM/yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out dataInicio) ||
        !DateTime.TryParseExact(fim, "dd/MM/yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out dataFim))
    {
        ViewBag.Error = "Formato de data inválido. Use o formato dd/MM/yyyy.";
        return View();
    }

    // Chamando o serviço para buscar as vendas
    var vendas = await _apiVendasService.GetVendasAsync(consola, dataInicio, dataFim);

    if (vendas == null)
    {
        ViewBag.Error = "Não foi possível obter os dados de vendas.";
        return View();
    }

    // Usando ViewBag para passar as datas de inicio e fim para a View
    ViewBag.DataInicio = dataInicio.ToString("dd/MM/yyyy");
    ViewBag.DataFim = dataFim.ToString("dd/MM/yyyy");

    return View("Resultados", vendas); // Passa a lista de vendas diretamente
}
```

Figura 11 - Controlador de vendas, para comunicação com a API.

A página web usa um serviço chamado `ApiVendasService` para se comunicar com a API. Esse serviço realiza uma chamada assíncrona ao endpoint da API utilizando a classe `HttpClient`, enviando as datas e o nome do console como parâmetros. Os dados retornados pela API são formatados e apresentados na interface de forma amigável, utilizando `ViewBag` para exibir as datas e os resultados das vendas.

```
4 referências
public class ApiVendasService
{
    private readonly HttpClient _httpClient;
    private readonly string _apiBaseUrl = "https://localhost:44369/api/vendas";

    0 referências
    public ApiVendasService(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }

    1 referência
    public async Task<List<ConsoleVendaDTO>> GetVendasAsync(string consola, DateTime inicio, DateTime fim)
    {
        var inicioString = inicio.ToString("yyyy-MM-dd");
        var fimString = fim.ToString("yyyy-MM-dd");
        var url = $"{_apiBaseUrl}/{consola}/periodo?inicio={inicioString}&fim={fimString}";

        var response = await _httpClient.GetAsync(url);

        if (response.IsSuccessStatusCode)
        {
            var jsonResponse = await response.Content.ReadAsStringAsync();
            return JsonConvert.DeserializeObject<List<ConsoleVendaDTO>>(jsonResponse);
        }

        return null;
    }
}
```

Figura 12 - Classe `ApiVendasService` que consome os dados da API.

A página web exibe os resultados das vendas para o console e o período selecionado pelo utilizador. Isso proporciona uma visão clara das vendas, oferecendo uma interface intuitiva e uma experiência de usuário agradável.

Dashboard Home Privacy Vendas

Consultar Vendas de Consolas

Escolha a Consola:

Playstation 3

Data de Inicio:

01/01/2000

Data de Fim:

01/01/2009

Buscar Vendas

Dashboard Home Privacy Vendas

Resultados de Vendas

Consulta realizada entre 01/01/2000 e 01/01/2009

Consola	Total de Vendas
PlayStation 3	80.96

Figura 13 - Busca e resultado em interface web.

7. Vídeo com demonstração (QR Code)

Para oferecer uma perspetiva mais abrangente e minuciosa sobre os resultados deste projeto, foi elaborado um vídeo explicativo. Neste vídeo, são exibidas as interfaces do ambiente de trabalho, que incluem o Pentaho, o MySQL, e os ficheiros gerados e o processo de envio de e-mails. A gravação proporciona uma demonstração visual da execução dos processos de integração de sistemas e ETL.

Acesso ao Vídeo:

Para visualizar o vídeo e assistir à demonstração completa do projeto, tem duas opções, ler o QR Code abaixo, ou então aceder diretamente o link, https://youtu.be/_1P66eDCSwI.

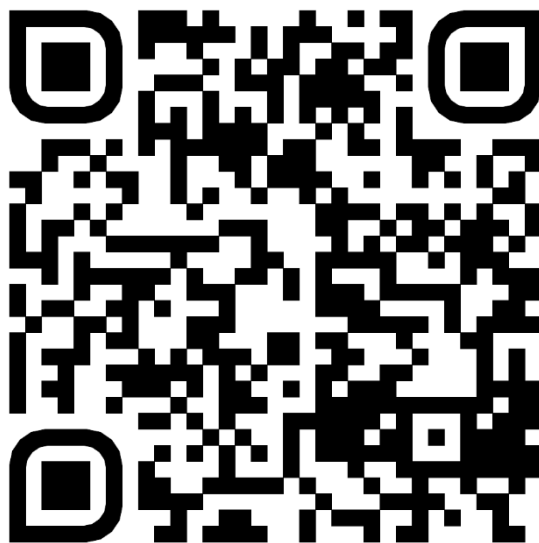


Figura 14 - QR Code que encaminha para vídeo.

8. Conclusão

Este projeto demonstrou, de maneira prática, a importância da integração de sistemas e da implementação de processos de **ETL (Extração, Transformação e Carga)** para a gestão eficaz de dados em uma organização. Ao conectar e automatizar o fluxo de dados de várias fontes e garantir que eles fossem transformados e validados corretamente, foi possível entregar informações de qualidade, alinhadas às necessidades do negócio.

Através do **Pentaho Data Integration (PDI)**, realizamos diversas tarefas que incluíram a extração de dados de arquivos CSV, a transformação desses dados em formatos compatíveis (como XML e MySQL) e a validação das informações utilizando ferramentas como validação de esquema XSD e verificação de tabelas e colunas em bases de dados. Também implementamos segurança através de criptografia AES para proteger os dados, além de enviar notificações automáticas por e-mail ao final de cada processo.

Além disso, a criação de uma **API** utilizando **ASP.NET Core** permitiu que os dados tratados fossem expostos de maneira eficiente para consumo por aplicações externas. A integração de uma **página web com arquitetura MVC** possibilitou ao utilizador final consultar e visualizar as vendas de consoles de videogame por diferentes períodos e categorias. Isso garantiu que a visualização dos dados fosse clara e precisa, possibilitando decisões baseadas em dados confiáveis.

Os principais benefícios do projeto foram:

1. **Automação e Eficiência:** O uso de processos automatizados, como ETL, eliminou a necessidade de manipulação manual de dados, economizando tempo e minimizando erros.
2. **Segurança de Dados:** A implementação de criptografia garantiu a proteção dos dados críticos durante o processamento e armazenamento.
3. **Validação e Confiabilidade:** A validação dos dados com esquemas XSD e a verificação da conformidade com as tabelas de banco de dados asseguraram a integridade e a consistência das informações.
4. **Escalabilidade e Flexibilidade:** A API desenvolvida permitiu consultas dinâmicas aos dados de vendas, oferecendo flexibilidade na forma de aceder e processar essas informações.
5. **Visibilidade e Controle:** A página web integrada ao sistema permitiu que os usuários finais acedem dados de vendas de forma simples e interativa, contribuindo para a transparência e eficiência no acompanhamento de resultados.

Este projeto mostrou como a combinação de ferramentas de ETL com o desenvolvimento de APIs e aplicações web pode transformar a maneira como as empresas lidam com grandes volumes de dados, garantindo que esses dados sejam úteis, precisos e acessíveis aos tomadores de decisão. Fiquei muito satisfeito com o projeto e com os desafios que enfrentei no seu desenvolvimento. O projeto atendeu às expectativas, me proporcionando mais competências e melhor conhecimento em uma ferramenta tão importante que é o Pentaho.

9. Bibliografia

<https://www.youtube.com/shorts/yyE2-N5ZjaM>

<https://www.youtube.com/watch?v=YgyYtbVpbXM&t=239s>

<https://www.youtube.com/watch?v=rLeC9iL0FOs&t=78s>

https://www.youtube.com/watch?v=vuyP2Uk_m7s

<https://www.youtube.com/watch?v=Boc23TgC4vE&t=51s>

<https://www.youtube.com/watch?v=Boc23TgC4vE&t=51s>

<https://www.youtube.com/watch?v=99yV9u3K9Dg>

<https://stackoverflow.com/questions/74644206/pentaho-why-does-xml-output-block-puts-out-csv>

<https://pipes.datavirtuality.com/connectors/visualize/flat-file-xml-csv-json-etc/pentaho/>

https://www.google.com/search?q=pentaho+mysql&oq=pentaho+mysql&gs_lcrp=EgZjaHJvbWUyBggAEFUYOTIGCAEQLhhA0gEINDEzN2owajmoAgCwAgA&sourceid=chrome&ie=UTF-8#

<https://docs.hitachivantara.com/r/en-us/pentaho-data-integration-and-analytics/9.3.x/mk-95pdia000/jdbc-drivers-reference/mysql>