

Trabalho ISI

Pentaho data integration

João Diogo Pinto Machado – 2115

## Índice do Relatorio

**No table of contents entries found.**

# Introdução ao tema

Nesta cadeira de ISI, Integracao de Sistemas Integrados, temos como professor o Prof. Lufer.

Assim ele apresentou-nos o trabalho para tratar principalmente do tema de ETL (Extract, Transform, Load) que é um processo essencial para gerir os dados que envolve apenas três etapas principais: extração, transformação e carregamento.

1. **Extração (Extract):** Nesta fase, os dados são coletados de diversas fontes, como bases de dados, APIs ou arquivos/ficheiros locais do computador como e o caso do meu trabalho. O objetivo é obter dados brutos que servirão para depois serem utilizados como base para o processo.
2. **Transformação (Transform):** Após a extração, os dados precisam ser convertidos, limpos para garantir que estejam no formato correto. Essa etapa envolve filtragem, agregação, normalização e a remoção de inconsistências.
3. **Carregamento (Load):** Na última etapa, os dados transformados são carregados em um sistema de armazenamento, como uma database local ou mesma na cloud, onde podem ser vistos para depois irem para análise e para mexer com elas de outra forma pretendida.

# O progama que utilizei

O **Pentaho Data Integration** (PDI), também conhecido como **Kettle**, é uma ferramenta de **ETL** (Extract, Transform, Load) usada para integrar, transformar e processar dados de diversas fontes. O PDI facilita a movimentação e a transformação de grandes volumes de dados de forma gráfica e intuitiva, sem a necessidade do código complexo que o normalmente acompanha.

**Principais Funcionalidades:**

* **Extração de Dados**: Conecta-se a uma ampla variedade de fontes de dados, como bases de dados SQL e NoSQL, arquivos planos (CSV, Excel), APIs, serviços web, entre outros.
* **Transformação de Dados**: Permite limpar, enriquecer, unir, filtrar e transformar dados. As transformações são feitas através de uma interface visual, com componentes arrastáveis (drag-and-drop).
* **Automação de Processos**: É possível agendar e monitorar fluxos de trabalho ETL recorrentes, gerando pipelines de dados automáticos.
* **Escalabilidade**: Suporta processamento em larga escala, possibilitando a integração de grandes volumes de dados, tanto localmente como quanto em ambientes distribuídos.

**Usos Comuns:**

* **Migração de dados**: Transferir dados de um sistema para outro.
* **Integração de fontes heterogêneas**: Consolidar dados de diferentes formatos e origens em uma única plataforma.

**Vantagens:**

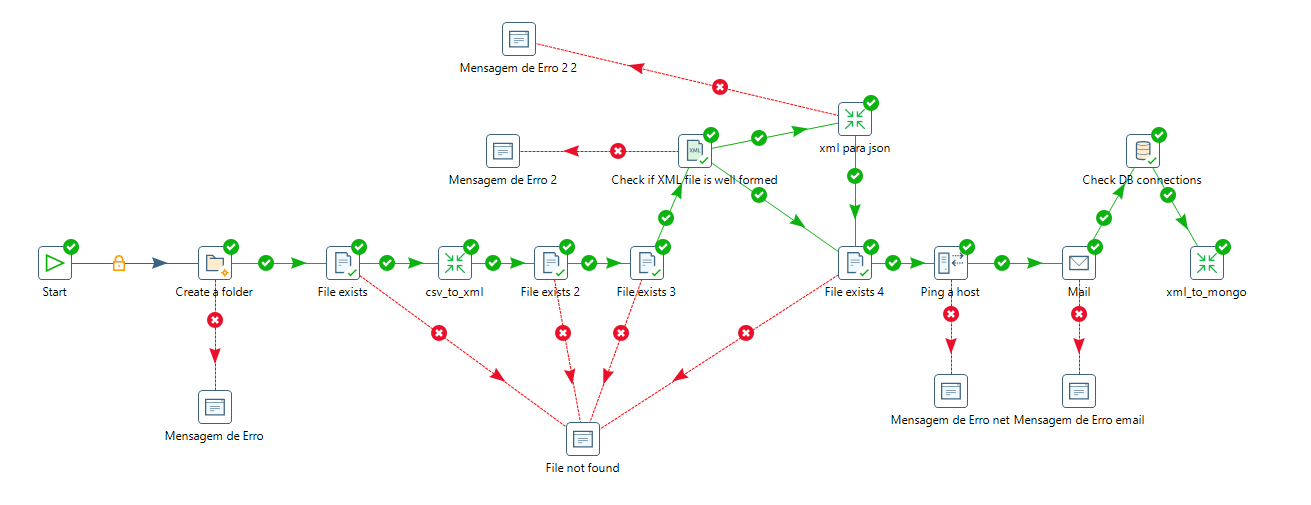
* **Interface gráfica amigável**: Não exige habilidades avançadas de programação.
* **Flexibilidade**: Suporta uma ampla gama de fontes de dados e formatos.
* **Automatização**: Cria processos ETL que podem ser facilmente agendados e monitorados.

Em resumo, o **Pentaho Data Integration** é uma solução poderosa e versátil para empresas e estudantes que precisam de consolidar e transformar dados de diversas fontes de forma eficiente e automática.

Esta mesma foi a opção que o professor abordou na aula, sendo ter sido ela a escolhida para este trabalho.

# O problema que abordei

O problema abordado aqui neste trabalho, foi o problema de ter um ficheiro num formato não desejado e fazer umas brincadeiras com o mesmo.



**Descrição do Problema Resolvido:**

Esse trabalho resolveu um problema típico de **ETL** que envolve:

* **Verificação de arquivos**: Se arquivos necessários existem e estão no formato correto.
* **Conversão de formatos**: Transformação de arquivos de **CSV para XML** e depois de **XML para JSON**.
* **Tratamento de erros**: Quando algum arquivo não é encontrado, uma mensagem de erro é acionada (File not found, Mensagem de Erro).
* **Envio para o MongoDB**: Após as transformações e validações, o fluxo envia os dados para o MongoDB.

# Para completar o trabalho

Para completar o trabalho iniciei o projeto de fazer uma API para visualizar melhor os dados da database, utilizando as varias ferramentas abaixo descritas:

**1. ASP.NET Core**

* **Framework**: Utilizado para construir a API em C#. Ele fornece uma estrutura robusta para criar APIs RESTful e lidar com requisições HTTP.
* **Ferramenta de Desenvolvimento**: Eu utilizei o Visual Studio da Microsoft, visto foi esta ferramenta que mais me habituei a trabalhar vindo de anos anteriores.
* **Recursos**: Controladores, autenticação, autorização, etc.

**2. MongoDB**

* **Banco de Dados NoSQL**: É a base de dados em que os dados da aplicação serão armazenados. MongoDB é conhecido por sua flexibilidade ao lidar com grandes volumes de dados em formato de documentos (JSON-like).
* **Ferramenta de Gestao**: MongoDB Compass (ferramenta gráfica), gerir e testar dados.

**3. MongoDB.Driver (Driver C# para MongoDB)**

* **Biblioteca oficial** do MongoDB para **.NET**, usada para fazer a integração entre a API e o MongoDB.
* **Operações CRUD**: Ele oferece métodos para **criar**, **ler**, **atualizar** e **deletar** documentos no MongoDB.

**4. Bibliotecas JSON (Newtonsoft.Json ou System.Text.Json)**

* Necessário para serializar e desserializar dados entre o MongoDB e a API.
* **Newtonsoft.Json**: Uma alternativa popular, usada para manipular JSON com mais opções de customização.

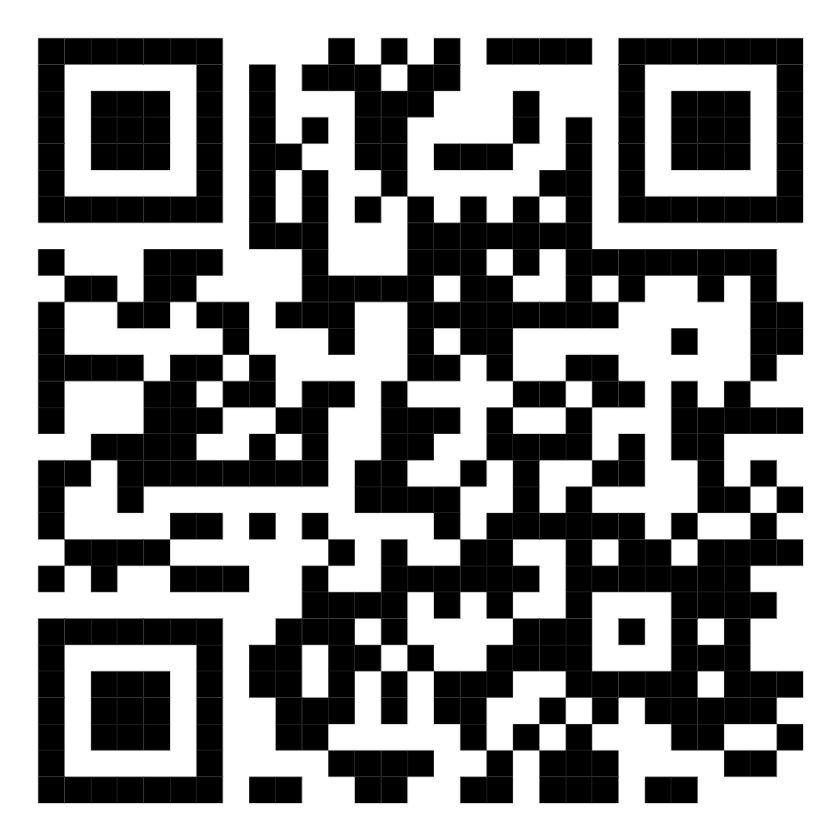
**5. Dependency Injection (DI)**

* ASP.NET Core usa **DI** para injetar o serviço de conexão com o MongoDB nos controladores.
* O serviço do MongoDB deve ser configurado uma única vez (como singleton) no arquivo Program.cs ou Startup.cs.

**Resumo do Processo:**

1. A **API** ASP.NET Core recebe uma requisição HTTP (por exemplo, um **GET** ou **POST**).
2. A **MongoDB.Driver** é usada para interagir com o MongoDB, fazendo operações CRUD no banco de dados.
3. A resposta é convertida de/para **JSON** usando **Newtonsoft.Json**.
4. O resultado da operação (sucesso ou erro) é retornado para o cliente via HTTP.

# Video do trabalho



# Em conclusão

Em conclusão, o trabalho realizado abrangeu a criação de um processo completo de integração de dados, utilizando o **Pentaho Data Integration** (PDI) para automação de processos de ETL e a implementação de uma API para interagir com o MongoDB. O uso do PDI permitiu a construção de um pipeline eficiente para extração, transformação e carga de dados, com tratamento de erros e validações em cada etapa do processo, garantindo a consistência e qualidade dos dados. Além disso, a tentativa de desenvolver a API para integrar a MongoDB demonstrou o esforço de criar uma solução complementar que facilita o gerenciamento dos dados processados.

O projeto como um todo proporcionou uma visão ampla das melhores práticas na manipulação e integração de dados em ambientes heterogêneos. A automação de tarefas, a verificação e validação de dados, bem como o foco em tratamento de erros e conectividade, garantiram que o processo no total fosse ágil, escalável e fácil de manter. Embora o tempo tenha sido limitado para finalizar a implementação da API, os resultados obtidos refletem a viabilidade da solução e abrem caminho para futuras melhorias e expansões na plataforma desenvolvida.

# Referências bibliográficas utilizadas

<https://www.mockaroo.com/>

<https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio>

<https://jsonlint.com/>

<https://docs.hitachivantara.com/r/en-us/pentaho-data-integration-and-analytics/9.5.x/mk-95pdia003> - Muito importante

<https://www.xmlvalidation.com/>

Obrigado.