ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

INSTITUTO POLITÉCNICO DO PORTO

Escola Superior de Tecnologia e Gestão

LICENCIATURA EM SEGURANÇA INFORMÁTICA EM REDES DE COMPUTADORES

Relatório de Integração de Dados Médicos: MedSync

Realizado por: Pedro Antunes *(8230068)* Ruben Nunes *(8230069)*

Unidade Curricular: Processamento Estruturado de Informação (PEI)

Janeiro 2025

1. Introdução

No atual cenário da saúde, a gestão eficiente de dados clínicos entre diferentes instituições hospitalares tornou-se um desafio crítico. O projeto `MedSync` surge como resposta à necessidade de estabelecer uma infraestrutura robusta para a partilha e integração de informações médicas entre hospitais parceiros (fictícios), tendo como tarefa melhorar a eficácia e qualidade dos cuidados prestados aos pacientes.

1.1 Objetivo do Projeto

Este projeto tem com principal objetivo desenvolver um sistema que permita a integração uniforme de dados médicos entre hospitais parceiros do consórcio `MedSync`. Para o tal, recorremos a implementação de um processo em três etapas fundamentais:

- Extração de Dados:
- Disponibilização via API REST
- Conversão para `XML` com `XQuery` (`BaseX`):

Adicionalmente, a padronização e partilha eficiente de dados clínicos permite:

- Melhor continuidade nos cuidados prestados aos pacientes
- Redução de custos operacionais através da eliminação de processos redundantes
- Tomada de decisões médicas mais informada e precisa perante os dados mensais
- Otimização da gestão de recursos hospitalares
- Maior eficiência no processo de transferência de pacientes entre instituições

---

1.2 Contexto

Dado o aumento da mobilidade dos pacientes entre diferentes hospitais, o `MedSync` pretende que todos os hospitais parceiros apresentem, mensalmente, um relatório consolidado sobre o histórico médico partilhado dos pacientes atendidos, incluindo diagnósticos, tratamentos realizados e transferências. Para isso, cada hospital terá de implementar um módulo no seu sistema informático para exportar a informação essencial relacionada com pacientes, atendimentos, tratamentos e transferências.

Para suportar a transferência padronizada de relatórios, decidiu-se criar um vocabulário `XML` comum que permita a integração uniforme dos dados. Cada hospital parceiro deverá implementar um módulo que suporte a geração de documentos `XML` (com informações de históricos médicos e transferências) de acordo com este vocabulário.

2. Contexto e Descrição

O `MedSync` é um consórcio de hospitais estabelecido com o objetivo fundamental de melhorar a continuidade e qualidade dos serviços de saúde através da partilha eficiente de dados clínicos. O sistema visa facilitar a troca de informações médicas entre instituições parceiras, garantindo que os dados dos pacientes sejam geridos de forma uniforme e segura.

2.1 Descrição e Caracterização

O sistema deve permitir que os hospitais parceiros apresentem mensalmente um relatório consolidado contendo:

- Histórico médico partilhado dos pacientes atendidos
- Diagnósticos realizados - Tratamentos executados
- Registos de transferências entre instituições

Como já mencionado previamente, o processo de integração de dados do `MedSync` é constituído por três etapas fundamentais:

2.2 Processo de Integração

2.2.1 Primeira Etapa

A primeira etapa do processo envolve a extração e normalização dos dados das bases de dados hospitalares:

- Os dados são extraídos das bases de dados hospitalares
- Armazenamento em `MongoDB` como base de dados intermediária
- Estruturação dos dados em coleções
- Mapeamento das estruturas de dados existentes
- Criação de índices para otimização de consultas

- Integração flexível de diferentes tipos de dados através de uma estrutura orientada a documentos

---

2.2.2 Segunda Etapa

A segunda etapa envolve a criação de uma camada de acesso padronizada através de uma API REST:

- Implementação de endpoints para acesso aos dados
- Formato `JSON` para facilitar a integração com diferentes sistemas
- Interface padronizada para consulta de informações
- Gestão de erros padronizada
- Respostas em formato `JSON` estruturado

2.2.3 Terceira Etapa

A etapa final envolve a **transformação dos dados** para o formato `XML` padronizado definido pelo `MedSync`:

- Definição do vocabulário `XML` comum
- Transformação dos dados `JSON` para o formato `XML`, sem ferir a ordem dos `JSON`
- Desenvolvimento de scripts `XQuery` para transformação
- Geração de relatórios estruturados segundo as especificações do `MedSync`
- Inclusão de metadados relevantes

3. Análise do Processo

3.1 Integração & Extração de Dados (Primeira Etapa)

A primeira etapa do processo de integração envolve a extração e normalização dos dados das bases de dados hospitalares. O `MongoDB` foi escolhido como sistema de armazenamento intermediário devido à sua natureza orientada a documentos, que oferece a flexibilidade necessária para integrar diferentes tipos de dados de forma estruturada e eficiente.

Os dados do sistema `MedSync` são extraídos de ficheiros `csv`, representando um subconjunto de informações hospitalares. Os ficheiros analisados foram:

Após análise detalhada dos requisitos, estabelecemos a seguinte abordagem para a etapa de extração:

- Os dados são extraídos das bases de dados dos hospitais e armazenados num `MongoDB` como base de dados intermediaria.
- Cada hospital armazena os seus dados de forma estruturada em coleções, garantindo maior organização e eficiência nas consultas do mesmo.

Ficheiro `csv`	**Descrição**
Pacientes.`csv`	Contém a lista de pacientes com os seus dados pessoais
RegistoClinico.`csv`	Inclui informações sobre cada atendimento médico realizado
Tratamentos.`csv`	Regista os tratamentos e prescrições de cada paciente
Transferencias.`csv`	Lista as transferências realizadas entre hospitais
Profissionais.`csv`	Contém informações sobre os profissionais de saúde envolvidos nos atendimentos
AtualizacaoTratamentos.`csv`	Representa as atualizações no estado dos tratamentos prescritos

Foram também criados índices para otimizar as consultas e melhorar o desempenho do sistema.

Os primeiros quatro ficheiros representam um **snapshot** completo de todos os dados acumulados até ao momento. No entanto, nas atualizações futuras:

- Apenas novos registos serão incluídos.
- O ficheiro `AtualizacaoTratamentos.csv` não inclui novos tratamentos, mas sim atualizações do estado de tratamentos já prescritos.

Essas informações foram fundamentais para modelar a estrutura de dados no `MongoDB` e definir o vocabulário `XML`.

Depois de muita discussão e debate, organizamos as coleções que nos foram fornecidas de forma que as mesmas cumprissem com os requisitos que nos foram propostos, destas quatro coleções, fizemos duas coleções que posteriormente irão ser usadas na API contendo só as informações necessárias

---

3.2 Implementação da API REST (Segunda Etapa)

Após a integração no `MongoDB`, os dados são disponibilizados em formato `JSON` através de uma API REST, para facilitar o acesso e manipulação dos dados.

A segunda etapa envolve a criação de uma camada de acesso padronizada através de uma API REST:

- Implementação de endpoints para acesso aos dados
- Formato `JSON` para facilitar a integração com diferentes sistemas
- Interface padronizada para consulta de informações
- Gestão de erros padronizada
- Respostas em formato `JSON` estruturado

Os dois endpoints links:

- `GET /api/registosClinicos?mes=XX&ano=YYYY` -> Obtém registos clínicos do mês/ano especificado
- `GET /api/transferencias?mes=XX&ano=YYYY` -> Obtém dados de transferências do mês/ano

A API REST permite que outros sistemas hospitalares consumam os dados de forma segura e eficiente, garantindo um fluxo contínuo de informação entre hospitais.

3.3 Conversão de Dados (Terceira Etapa)

Para garantir que os relatórios sejam gerados no formato `XML` padrão definido pelo `MedSync`, utiliza-se o `BaseX`, que permite converter os dados `JSON` para `XML` utilizando scripts `XQuery`. Este processo assegura que os relatórios atendam às exigências do vocabulário `XML` comum.

A etapa final envolve a transformação dos dadosnpara o formato `XML` padronizado definido pelo `MedSync`:

- Definição do vocabulário `XML` comum
- Transformação dos dados `JSON` para o formato `XML`, sem ferir a ordem dos `JSON`
- Desenvolvimento de scripts `XQuery` para transformação
- Geração de relatórios estruturados segundo as especificações do `MedSync`
- Inclusão de metadados relevantes

Este processo assegura que todas as instituições hospitalares sigam um formato de dados uniforme, facilitando a integração entre diferentes sistemas.

3.4 Identificação de dados omissos e justificação das decisões tomadas

Durante a análise dos ficheiros `csv`, identificámos alguns **dados omissos** e definimos estratégias para lidar com essas lacunas:

Dado Omissos	Solução Adotada
Telefones ou e-mails ausentes	Substituir valores ausentes por "Não fornecido" no `XML`
Especialidade do atendimento em falta	Inferida com base no profissional de saúde responsável.
Ficheiro `Tratamentos.csv` sem ligação com os restantes ficheiros	Substituição no ficheiro `RegistoClinico.csv` o campo `ID_Atendimento` por `ID_Registo_Clinico` para ter uma relação com o ficheiro `Tratamentos.csv`
Faixa Etária em falta	Adicionada a faixa etária no paciente atendendo a sua data de nascimento
Tipo de paciente em falta	Adicionado o `Tipo_Paciente` consoante certos requisitos escritos no enunciado

Estas decisões garantem que o `XML` **gerado seja sempre válido**, evitando a introdução de **dados inconsistentes ou incompletos**.

A definição do **vocabulário** `XML`, as **restrições aplicadas** e a **análise dos ficheiros** `csv` foram essenciais para garantir a **uniformização dos dados clínicos** no `MedSync`. O processo estruturado de **extração, transformação e armazenamento** assegura que os hospitais parceiros possam **partilhar e consultar os dados médicos de forma eficiente e segura**.

4. Exemplos Reais de Sistemas Similares de Integração Hospitalar

---

Para termos uma noção mais clara do trabalho, realizamos uma breve pesquisa para ver se encontrávamos sistemas similares para termos uma ideia mais clara sobre o tema. Com essa pesquisa descobrimos que em Portugal, existem **diversos sistemas de informação hospitalar** que visam facilitar essa integração e garantir o fluxo contínuo de dados clínicos entre unidades de saúde. Entre elas, destacam-se o **SClínico Hospitalar** e o **SONHO(Sistema Integrado de Informação Hospitalar)**.

4.1 SClínico Hospitalar

O **SClínico Hospitalar** é um **sistema de registo clínico eletrônico** desenvolvido pela **SPMS (Serviços Partilhados do Ministério da Saúde)**. Este sistema é amplamente utilizado nas unidades hospitalares do **Serviço Nacional de Saúde (SNS)** e tem como principal objetivo **unificar e centralizar os registos clínicos dos pacientes**, garantindo a sua partilha entre os diferentes profissionais de saúde. A utilização do **SClínico Hospitalar** melhora a **eficiência dos serviços de saúde**, permitindo que os profissionais tenham **acesso rápido e seguro às informações clínicas**otimizando a continuidade dos cuidados prestados.

4.2 SONHO (Sistema Integrado de Informação Hospitalar)

O **SONHO** é um dos sistemas mais antigos e fundamentais no SNS, sendo utilizado por **cerca de 90% das instituições hospitalares públicas** em Portugal. Desenvolvido para **gerir dados administrativos e clínicos dos utentes**, este sistema funciona como a **base para a troca de informações entre hospitais e outras unidades de saúde**.

O **SONHO** é essencial para a **gestão operacional dos hospitais** e para a **integração com outras plataformas de registo clínico**, sendo uma peça-chave na interligação entre diferentes unidades hospitalares.

4.3 Justificação da Relevância do Projeto

Com bases das pesquisas e nos exemplos reais estudados, podemos dizer que a **integração de dados médicos** é essencial para garantir a **continuidade e eficiência no tratamento de pacientes**. Sem um sistema unificado, diversos problemas podem surgir:

1. Evitar Perda de Informações Clínicas

- Sem um sistema integrado, há **riscos de perda de dados** importantes durante a transferência de pacientes entre hospitais
- Diagnósticos e tratamentos podem **não ser partilhados corretamente**, resultando em atrasos e falhas no atendimento

2. Facilidade de Acesso e Melhor Planeamento

- O acesso rápido e padronizado aos registos clínicos melhora a tomada de decisões médicas
- A análise dos dados permite que os hospitais realizem **planeamento estratégico**, alocando melhor os recursos

5. Requisitos e Especificações

Esta secção detalha os requisitos necessários para a estruturação e integração dos dados no `MedSync`, especificando os limites dos elementos `XML`, as regras de validação e a análise dos dados fornecidos nos ficheiros `csv`.

5.1 Detalhamento dos requisitos vocabulário XML

Para garantir a uniformização e padronização dos dados entre os hospitais parceiros, foi definido um **vocabulário** `XML` **comum** que contém todas as informações essenciais para os relatórios mensais. A estrutura do `XML` inclui os seguintes elementos principais

---

5.2 Informações Gerais do Hospital

Cada relatório `XML` deve conter informações básicas sobre o hospital que está a gerar o documento, no nosso caso definimos este padrão:

- Elemento principal (`<HospitalInfo>`)
  - Código único do hospital (`<CodigoHospital>`)
  - Nome do hospital (`<Nome>`)
  - Morada do hospital (`<Morada>`)
  - Mês e ano do relatório (`<DataRelatorio>`)

Pseudocodigo XML

```
<HospitalInfo>
        <CodigoHospital>...</CodigoHospital>
        <Nome>...</Nome>
        <Morada>...</Morada>
        <DataRelatorio>...</DataRelatorio>
</HospitalInfo>
```

5.3 Registos Clínicos

Os registos clínicos devem conter informações sobre os atendimentos realizados:

- Registos clínicos do mês/ano (`<RegistoClinicos>`)
  - Um Registo Clinico (`<RegistoClinico>`) com as seguintes informações:
    - Identificação do atendimento (`<ID_Atendimento>`)
    - Identificação do paciente (`<ID_Paciente`)
    - Identificação do profissional (`<ID_Profissional>`)
    - Data e hora do atendimento (`<Data_Atendimento>`)
    - Dados do profissional de saúde responsável (`<Responsável>`)
      - Identificação do profissional (`<ID_Profissional>`)
      - Nome do profissional (`<Nome_Completo>`)
      - Especialidade do profissional (`<Especialidade>`)
    - Tipo do diagnostico (`<Tipo_Diagnostico>`)
    - O código referente ao problema de saúde (`<Codigo_CID10>`)
    - Descrição do diagnostico (`<Descricao_Diagnostico>`)
    - Todos os tratamentos feitos (`<Tratamentos>`)
      - Identificação do tratamento (`<ID_Tratamento>`)
      - Identificação do paciente (`<ID_Paciente>`)
      - Identificação do registo clinico (`<ID_Registo_Clinico>`)
      - Tipo de tratamento realizado (`<Tipo_Tratamento>`)
      - Se o tratamento foi realizado ou não (`<Realizado>`)
    - Dados relativos ao paciente (`<Paciente>`)
      - Nome do paciente (`<Nome_Completo>`)
      - Data de nascimento do paciente (`<Data_Nascimento>`)
      - Faixa etária do paciente (`<Faixa_Etaria>`)
      - Género (`<Género>`)
      - Telefone do paciente (`<Telefone>`)
      - Email do paciente (`<Email>`)
      - Identificação do paciente (`<ID_Paciente`)

Pseudocodigo em XML

```
<RegistosClinicos mes="12" ano="2024">
        <RegistoClinico>
        <!-- Dados do Atendimento -->
                <ID_Atendimento>...</ID_Atendimento>
                <Data_Atendimento>...</Data_Atendimento>
                <ID_Paciente>...</ID_Paciente>

                <!-- Dados do Responsável -->
                <Responsavel>
                        <ID_Profissional>...</ID_Profissional>
                        <Nome_Completo>...</Nome_Completo>
                        <Especialidade>...</Especialidade>
                </Responsavel>

                <!-- Dados do Diagnóstico -->
                <Tipo_Diagnostico>...</Tipo_Diagnostico>
                <Codigo_CID10>...</Codigo_CID10>
                <Descricao_Diagnostico>...</Descricao_Diagnostico>

                <!-- Dados dos Tratamentos -->
                <Tratamentos>
                        <Tratamento>
                                <ID_Tratamento>...</ID_Tratamento>
                                <Tipo_Tratamento>...</Tipo_Tratamento>
                                <Realizado>...</Realizado>
                        </Tratamento>
                </Tratamentos>

                <!-- Dados do Paciente -->
                <Paciente>
                        <Nome_Completo>...</Nome_Completo>
                        <Data_Nascimento>...</Data_Nascimento>
                        <Faixa_Etaria>...</Faixa_Etaria>
                        <Genero>...</Genero>
                        <Telefone>...</Telefone>
                        <Email>...</Email>
                </Paciente>

        </RegistoClinico>
</RegistosClinicos>
```

5.4 Registos Clínicos Mensais

Deve ser gerado um resumo mensal contendo estatísticas gerais do hospital:

- Registos mensais (`<RegistosMensais>`)
  - Número de pacientes faixa etária/gênero(`<PacientesPorFaixaEtariaGenero>`)
    - Um grupo por faixa etária/gênero (`<FaixaEtariaGenero>`)
      - Total de pacientes (`<TotalPacientes>`)
      - Faixa etária (`<FaixaEtaria>`)
      - Gênero (`<Genero>`)
  - Número de casos por especialidade (`<CasosPorEspecialidade>`)
    - Uma especialidade médica (`<Especialidade>`)
    - Nome da especialidade (`<Nome>`)
    - Total de casos atendidos (`<Casos>`)
  - Número total de tratamentos realizados (`<NumeroTratamentosRealizados>`)
  - Número total doenças crônicas (`<NumeroCondicoesCronicasDiagnosticadas>`)

Pseudocodigo em XML

```xml
<RegistosMensais mes="12" ano="2024">
   <!-- Estatísticas por Faixa Etária e Género -->
   <PacientesPorFaixaEtariaGenero>
       <FaixaEtariaGenero>
                        <TotalPacientes>...</TotalPacientes>
                <FaixaEtaria>...</FaixaEtaria>
                <Genero>...</Genero>
       </FaixaEtariaGenero>
   </PacientesPorFaixaEtariaGenero>

   <!-- Casos por Especialidade Médica -->
   <CasosPorEspecialidade>
       <Especialidade>
           <Nome>Cardiologia</Nome>
           <Casos>234</Casos>
       </Especialidade>
       <!-- Outras especialidades... -->
   </CasosPorEspecialidade>

   <!-- Totais Gerais -->
   <NumeroTratamentosRealizados>1234</NumeroTratamentosRealizados>

<NumeroCondicoesCronicasDiagnosticadas>...
</NumeroCondicoesCronicasDiagnosticadas>

</RegistosMensais>
```

5.5 Transferências

As transferências entre hospitais devem ser detalhadas, incluindo informações sobre o hospital de destino e os motivos da transferência:

- Transferências (`<Transferencias>`)
  - Uma transferência específica (`<Transferencia>`)
    - Identificação da transferência (`<ID_Transferencia>`)
    - Identificação do paciente (`<ID_Paciente>`)
    - Data da transferência (`<Data_Transferencia>`)
    - Motivo da transferência (`<Motivo>`)
    - Destino da transferência (`<Destino>`)
    - Tipo da transferência (`<Tipo_Transferencia>`)
    - Hospital de origem (`<Hospital_Origem>`)
    - Relatório médico (`<Relatorio_Medico>`)
      - Conteúdo do relatório (`<Relatorio>`)
      - Identificação do atendimento (`<ID_Atendimento>`)
      - Data do atendimento (`<Data_Atendimento>`)
      - Responsável pelo atendimento (`<Responsavel>`)
        - Identificação do profissional (`<ID_Profissional>`)
        - Nome do profissional (`<Nome_Completo>`)
        - Especialidade do profissional (`<Especialidade>`)
      - Diagnóstico (`<Diagnostico>`)
      - Tipo do diagnóstico (`<Tipo_Diagnostico>`)
      - Código CID-10 do diagnóstico (`<Codigo_CID10>`)
      - Descrição do diagnóstico (`<Descricao_Diagnostico>`)
      - Indicação se é um diagnóstico crônico (`<Diagnostico_Cronico>`)
      - Tratamentos realizados (`<Tratamentos>`)
        - Um tratamento específico (`<Tratamento>`)
        - Identificação do tratamento (`<ID_Tratamento>`)
        - Tipo do tratamento realizado (`<Tipo_Tratamento>`)
        - Indicação se o tratamento foi realizado (`<Realizado>`)

Pseudocodigo em XML

```xml
<Transferencias mes="12" ano="2024">
   <Transferencia>
       <!-- Informações Básicas da Transferência -->
            <ID_Transferencia>...</ID_Transferencia>
                <ID_Paciente>...</ID_Paciente>
            <Data_Transferencia>...</Data_Transferencia>
            <Motivo>...</Motivo>
                <Destino>...</Destino>
            <Tipo_Transferencia>...</Tipo_Transferencia>
                <Hospital_Origem>...</Hospital_Origem>

       <!-- Relatório Médico Detalhado -->
       <Relatorio_Medico>
                <Relatorio>
                    <ID_Atendimento>...</ID_Atendimento>
                    <Data_Atendimento>...</Data_Atendimento>

                <!-- Profissional Responsável -->
                    <Responsavel>
                        <ID_Profissional>...</ID_Profissional>
                        <Nome_Completo>...</Nome_Completo>
                        <Especialidade>...</Especialidade>
                    </Responsavel>

                <Tipo_Diagnostico>...</Tipo_Diagnostico>
                <Codigo_CID10>...</Codigo_CID10>
                <Descricao_Diagnostico>...</Descricao_Diagnostico>
                <Diagnostico_Cronico>...</Diagnostico_Cronico>

                                <!-- Tratamentos Realizados -->
                    <Tratamentos>
                        <Tratamento>
                            <ID_Tratamento>...</ID_Tratamento>
                            <Tipo_Tratamento>...</Tipo_Tratamento>
                            <Realizado>Sim</Realizado>
                        </Tratamento>
                    </Tratamentos>
                </Relatorio>
            </Relatorio_Medico>
        </Transferencia>
</Transferencias>
```

5.6 Transferências Mensais

Deve ser gerado um resumo mensal contendo estatísticas gerais das transferências:

- Relatório de Transferências Mensais (`<RelatorioTransferenciasMensais>`)
- Total de transferências realizadas no mês (`<TotalTransferencias>`)
- Transferências por motivo (`<TransferenciasPorMotivo>`)
  - Um motivo específico (`<Motivo>`)
    - Descrição do motivo (`<Descricao>`)
    - Total de transferências relacionadas a este motivo (`<Total>`)
  - Transferências por tipo (`<TransferenciasPorTipo>`)
    - Um tipo específico (`<Tipo>`)
      - Descrição do tipo de transferência (`<Descricao>`)
      - Total de transferências desse tipo (`<Total>`)
  - Transferências por hospital (`<TransferenciasPorHospital>`)
    - Um hospital específico (`<Hospital>`)
      - Nome do hospital (`<Nome>`)
      - Total de transferências realizadas para esse hospital (`<Total>`)

Pseudocodigo em XML

```
<RelatorioTransferenciasMensais mes="12" ano="2024">
        <!-- Total Geral de Transferências -->
        <TotalTransferencias>256</TotalTransferencias>

        <!-- Agrupamento por Motivo -->
        <TransferenciasPorMotivo>
            <Motivo>
                <Descricao>...</Descricao>
                <Total>...</Total>
            </Motivo>
        </TransferenciasPorMotivo>

        <!-- Agrupamento por Tipo -->
        <TransferenciasPorTipo>
            <Tipo>
                        <Descricao>...</Descricao>
                <Total>...</Total>
            </Tipo>
        </TransferenciasPorTipo>

   <!-- Agrupamento por Hospital -->
        <TransferenciasPorHospital>
            <Hospital>
                <Nome>...</Nome>
                <Total>...</Total>
            </Hospital>
        </TransferenciasPorHospital>
</RelatorioTransferenciasMensais>
```

6. Modelação de Dados em MongoDB

A modelação de dados no `MongoDB` para o projeto `MedSync` foi desenvolvida considerando três:

- Sequencias de acesso aos dados
- Relacionamentos entre as entidades (coleções) envolvidas
- O evitar de "overhead/unwind" ao recorrer a certas coleções

---

6.1 Estratégia de Estruturação dos Dados

A estruturação dos dados seguiu uma abordagem, equilibrando a **normalização e desnormalização**, com o objetivo de otimizar a **eficiência de consultas** e a **gestão dos dados clínicos**.

As decisões tomadas nesta fase foram baseadas nos seguintes princípios:

1. Incorporação de documentos (`Document Embedding`)

- Aplicado para dados frequentemente acedidos em conjunto, como informações do paciente dentro dos registos clínicos.

2. Utilização do `Computed Pattern`

- Implementado para armazenar campos calculados frequentemente consultados, especialmente para a geração de relatórios mensais.

3. Aplicação do `Subset Pattern`

- Usado para otimizar o acesso a grandes volumes de dados, permitindo consulta eficiente ao histórico sem carregar informações desnecessárias.

6.2 Justificação da Abordagem Escolhida

A abordagem de **modelação de dados no** `MongoDB` foi definida com o objetivo de garantir **eficiência, escalabilidade e desempenho otimizado**, considerando a necessidade de consultas frequentes sobre pacientes, registos clínicos e transferências.

A modularização escolhida seguiu **três princípios chave** para organizar os dados de forma eficiente:

1. Document Embedding (Incorporação de Dados)

- Utilizado para **dados que são frequentemente acedidos em conjunto**, reduzindo a necessidade de consultas (`$lookup`) complexas.

  Onde foi aplicado?
  - Dentro dos **Registos Clínicos**, incorporamos informações sobre os **Tratamentos**, **Responsável Médico** e informações sobre o **Paciente**.
  - Dentro das **Transferências**, incorporamos o **Histórico Clínico do Paciente** até a data da transferência.

Vantagens:

- Menos junções complexas (`$lookup`), facilitando as **consultas mais rápidas**.
- **Menos chamadas à base de dados**, reduzindo o tempo de resposta da API.
- Garante **consistência dos dados** dentro de uma única coleção.

2. Computed Pattern (Campos Pré-Calculados)

- Foi aplicada para armazenar **dados que são frequentemente consultados e calculados**. Em vez de processar os dados a cada consulta, armazenamos os valores já processados para otimizar a performance.

  Onde foi aplicado?
  - **Cálculo da faixa etária dos pacientes**.
  - **O tipo de paciente**.
  - **Se o diagnostico era crônico ou não**.

Vantagens:

- Evita **cálculos repetitivos em tempo real**, reduzindo o processamento da API.

Com esta abordagem, conseguimos um **balanço eficiente entre desempenho, escalabilidade e facilidade de acesso aos dados**, garantindo que a modelação do `MedSync` seja **otimizado para o uso real em ambiente hospitalar**.

6.3 Estrutura das Coleções (Originais/Extraídas)

Análise dos Dados de Origem Os dados fornecidos em formato `csv` representam um subconjunto de informações típicas de um hospital sendo organizadas no `MongoDB` conforme descrito abaixo.

Ficheiro `csv`	**Descrição**
Pacientes.`csv`	Contém a lista de pacientes com os seus dados pessoais
RegistoClinico.`csv`	Inclui informações sobre cada atendimento médico realizado
Tratamentos.`csv`	Regista os tratamentos e prescrições de cada paciente
Transferencias.`csv`	Lista as transferências realizadas entre hospitais
Profissionais.`csv`	Contém informações sobre os profissionais de saúde envolvidos nos atendimentos
AtualizacaoTratamentos.`csv`	Representa as atualizações no estado dos tratamentos prescritos

---

6.3.1 Pacientes.csv

Contém a lista de pacientes com os seus dados pessoais

_id	ObjectID
ID_Paciente	Number
Nome_Completo	String
Data_Nascimento	Date
Género	String
Telefone	String
Email	String
Data_Registo	Date

```
// Pacientes.csv
{
        _id: ObjectID,
        ID_Paciente: Number
        Nome_Completo: String,
        Data_Nascimento: Date,
        Genero: String,
        Telefone: String,
        Email: String,
        Data_Registo: Date,
}
```

Indices: `ID_Paciente`

6.3.2 RegistosClinicos.csv

Inclui informações sobre cada atendimento médico realizado

_id	ObjectID
ID_Atendimento	Number
ID_Paciente	Number
ID_Profissional	Number
Data_Atendimento	Date
Tipo_Diagnostico	String
Codigo_CID10	String
Descricao_Diagnostico	String

```
// RegistosClinicos.csv
{
        _id: ObjectID,
        ID_Atendimento: Number,
        ID_Paciente: Number
        ID_Profissional: Number,
        Data_Nascimento: Date,
        Tipo_Diagnostico: String,
        Codigo_CID10: String,
        Descricao_Diagnostico: String,
}
```

Indices: `ID_Atendimento`, `ID_Paciente`, `ID_Profissional`

6.3.3 Tratamentos.csv

Regista os tratamentos e prescrições de cada paciente

_id	ObjectID
ID_Tratamento	Number
ID_Registo_Clinico	Number
Tipo_Tratmento	String
Realizado	String

```
// Tratamentos.csv
{
        _id: ObjectID,
        ID_Tratmento: Number,
        ID_Registo_Clinico : Number,
        Tipo_Tratamento: String,
        Realizado: String,
}
```

Indices: `ID_Registo_Clinico`, `ID_Tratamento`

6.3.4 Transferencias.csv

Lista as transferências realizadas entre hospitais

_id	ObjectID
ID_Transferencia	Number
ID_Paciente	Number
ID_Profissional	Number
Data_Transferencia	Date
Motivo	String
Destino	String
Tipo_Transferencia	String
Hospital_Destino	String

```
// Transferencias.csv
{
        _id: ObjectID,
        ID_Transferencia: Number,
        ID_Paciente: Number,
        ID_Profisional: Number,
        Data_Transferencia: Date,
        Motivo: String,
        Destino: String,
        Tipo_Transferencia: String,
        Hospital_Destino: String,
}
```

Indices: `ID_Transferencia`, `ID_Paciente`, `ID_Profissional`

6.3.5 Profissionais.csv

Contém informações sobre os profissionais de saúde responsáveis pelos atendimentos.

_id	ObjectID
ID_Profissional	Number
Nome_Completo	String
Especialidade	String
Anos_Experiencia	Number

```
// Profissionais.`csv`
{
        _id: ObjectID,
        ID_Profisional: Number,
        Nome_Completo: String,
        Especialidade: String,
        Anos_Experiencia: String,
}
```

Indicies: `ID_Profissional`

6.3.6 AtualizacaoTratamentos.csv

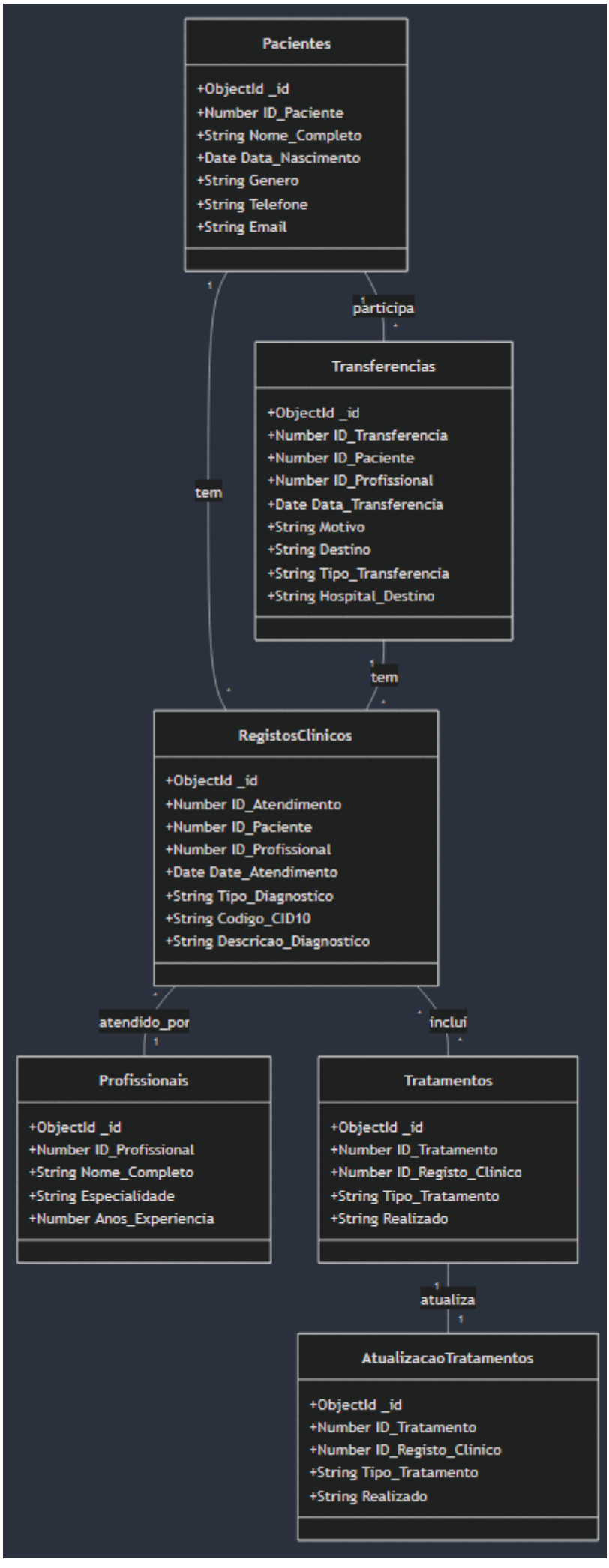
Representa as atualizações no estado dos tratamentos prescritos

_id	ObjectID
ID_Tratamento	Number
ID_Registo_Clinico	Number
Tipo_Tratmento	String
Realizado	String

```
// AtualizacaoTratamentos.`csv`
{
        _id: ObjectID,
        ID_Tratmento: Number,
        ID_Registo_Clinico : Number,
        Tipo_Tratamento: String,
        Realizado: String,
}
```

6.4 Abordagem da Modelação de Coleções

A nossa abordagem e interpretação dos dados extraídos foi principalmente estabelecida através de da descrição das exigências do enunciado e dos relacionamentos entre as coleções importadas do `.csv` `para o` MongoDB`` onde chegamos a justificar um total de seis relacionamentos entre as entidades envolvidas.


Pacientes
+ObjectId _id
+Number ID_Paciente
+String Nome_Completo
+Date Data_Nascimento
+String Genero
+String Telefone
+String Email
1
participa
1
Transferencias
+ObjectId _id
+Number ID_Transferencia
+Number ID_Paciente
+Number ID_Profissional
+Date Data_Transferencia
+String Motivo
+String Destino
+String Tipo_Transferencia
+String Hospital_Destino
tem
1
tem
RegistosClinicos
+ObjectId _id
+Number ID_Atendimento
+Number ID_Paciente
+Number ID_Profissional
+Date Date_Atendimento
+String Tipo_Diagnostico
+String Codigo_CID10
+String Descricao_Diagnostico
atendido_por
1
inclui
Profissionais
+ObjectId _id
+Number ID_Profissional
+String Nome_Completo
+String Especialidade
+Number Anos_Experiencia
Tratamentos
+ObjectId _id
+Number ID_Tratamento
+Number ID_Registo_Clinico
+String Tipo_Tratamento
+String Realizado
1
atualiza
1
AtualizacaoTratamentos
+ObjectId _id
+Number ID_Tratamento
+Number ID_Registo_Clinico
+String Tipo_Tratamento
+String Realizado

6.4.1 Interpretação da Modelação

Curta interpretação dos relacionamentos:

- Um `Paciente` pode ter vários `RegistosClinicos` (1:N)
- Um `Paciente` participa em várias `Transferencias` (1:N)
- Um `Profissional` faz atendimento de vários `RegistosClinicos` (1:N)
- Uma `Transferencia` tem um histórico de vários `RegistosClinicos` (1:N)
- Vários `RegistosClinicos` contém vários `Tratamentos` (N:N)
- Um `Tratamento` leva uma atualização de um `AtualizacaoTratamentos` (1:1)

Pacientes <-> RegistosClinicos:

- Relacionamento através do (`ID_Paciente`)
- Permite consulta através de um `Paciente` verificar o seu histórico de `Registos_Clinicos`

```
$lookup: {
    from: 'RegistosClinicos',
    localField: 'ID_Paciente',
    foreignField: 'ID_Paciente',
    as: 'Registos_Clinicos'
}
```

Transferencias <-> RegistosClinicos:

- Relacionamento através do (`ID_Paciente`)
- Ajuda no restreamento do `Relatorio_Medico` e dados de um `Paciente` até á `Data_Transferencia` de uma `Transferencias`

```
$lookup: {
    from: 'RegistosClinicos',
    localField: 'ID_Paciente',
    foreignField: 'ID_Paciente',
    as: 'Relatorio_Medico'
}
```

RegistosClinicos <-> Profissionais

- Relacionamento através do (`ID_Profissional`)
- Permite a associação do `Responsavel` de sáude por cada `RegistosClinicos`

```
$lookup: {
    from: 'Profissionais',
    localField: 'ID_Profissional',
    foreignField: 'ID_Profissional',
    as: 'Responsavel'
}
```

RegistosClinicos <-> Tratamentos

- Relacionamento através do (`ID_Registo_Clinico`)
- Facilita a integração dos `Tratamentos` (sejam eles prescritos, realizados ou não realizados) dentro de um `RegistosClinicos`

```
$lookup: {
        from: 'Tratamentos',
        localField: 'ID_Registo_Clinico',
        foreignField: 'ID_Registo_Clinico',
        as: 'Tratamentos'
}
```

Tratamentos <-> AtualizacaoTratamentos

- Relacionamento simples e único através do (`ID_Tratamento`)
- Traz flexibilidade em atualizar o snapshot `Tratamentos` com a coleção `AtualizaçãoTratamentos` que contém o estado de `Tratamentos` recente de um `Pacientes`

```
$lookup: {
        from: 'AtualizacaoTratamentos',
        localField: 'ID_Tratamento',
        foreignField: 'ID_Tratamento'
        as: 'Tratamentos_Atualizados' // Que passa para Tratamentos
                                                                // para
evitar redundancia.
}
```

6.4.2 Modelação de Dados dos Pacientes

Cada `Paciente` deverá conter informações detalhadas respetivo a cada um eles com o seguinte:

- Número de identificação único (`ID_Paciente`)
- Nome completo (`Nome_Completo`)
- Data de nascimento (`Data_Nascimento`)
- Idade (`Idade`)
- Faixa Etária relevante para estatísticas (`Faixa_Etaria`)
- Género (`Género`)
- Contactos do individuo (`Contactos`)
  - do Telefone (`Contactos.Telefone`)
  - do Email (`Contactos.Email`)
- Tipo de Paciente (`Tipo_Paciente`)
- Histórico de atendimentos (`Registos_Clinicos`)
  - Com diagnósticos principais (`Registos_Clinicos.Principais`)
  - Com diagnósticos secundários (`Registos_Clinicos.Secundários`)
  - Com diagnósticos crónicos (`Registos_Clinicos.Crónicos`)
- Histórico de tratamentos (`Registos_Tratamentos`)
  - Com tratamentos realizados (`Registos_Tratamentos.Realizados`)
  - Com tratamentos não realizados (`Registos_Tratamentos.Nao_Realizados`)
- Histórico de transferências (`Registos_Transferencias`)
  - Com transferências urgentes (`Registos_Transferencias.Urgentes`)
  - Com transferências eletivas (`Registos_Transferencias.Eletivas`)

_id	ObjectID
ID_Paciente	Number
Nome_Completo	String
Data_Nascimento	Date
Idade	Number
Faixa_Etaria	String
Género	String
Contacto	Object
Contacto.Telefone	String
Contacto.Email	String
Tipo_Paciente	String
Registos_Clinicos	Object
Registos_Clinicos.Principais	Registos_Clinicos[ ]
Registos_Clinicos.Secundarios	Registos_Clinicos[ ]
Registos_Clinicos.Cronicos	Registos_Clinicos[ ]
Registos_Tratamentos	Object
Registos_Tratamentos.Realizados	Tratamentos[ ]
Registos_Tratamentos.Nao_Realizados	Tratamentos[ ]
Registos_Transferencias	Object
Registos_Transferencias.Urgentes	Transferencias[ ]
Registos_Transferencia.Eletivas	Transferencias[ ]

```
{
  _id: ObjectID,
  ID_Paciente: Number,
  Nome_Completo: String,
  Data_Nascimento: Date,
  Idade: Number,
  Faixa_Etaria: String,
  Genero: String,

  // Contactos do Paciente
  Contacto: {
    Telefone: "String",
    Email: "String",
  },

  Tipo_paciente: String,

  // Histórico de atendimentos (RegistoClinicos)
  // do Paciente nos quais foram...
  Registos_Clinicos : {
    // ...diagnosticados como principal.
        Principais: [{ ... }]
        // ...diagnosticados como secundario.
    Secudarios: [{ ... }]
        // ...diagnosticados como cronico
        Cronicos: [{ ... }]
   },

  // Histórico de tratamentos (Tratamentos)
  // do Paciente nos quais foram...
  Registos_Tratamentos: {
    // ...validados como realizados
        Realizados: [{ ... }],
        // ...validados como nao realizados
    Nao_Realizados: [{ ... }]
  },

  // Histórico de transferencias ocurridas (Transferencias)
  // com o Paciente nos quais foram...
  Registos_Transferncias: {
    // ...efetuadas com urgencia
        Urgentes: [{ ... }],
    // ...efetuadas com planeamento
        Eletivas: [{ ... }]
  },
}
```

A modelação de dados para o sistema `MedSync` foi desenvolvida considerando os cenários de acesso de os utilizadores futuros e requisitos de performance. No que diz respeito aos dados dos pacientes, optou-se por uma abordagem de incorporação (embedding) para informações frequentemente acedidas em conjunto, como dados de contacto, condições crónicas e históricos de tratamentos. Esta decisão reduz significativamente o número de operações de junção necessárias, melhorando o acesso ao utilizador e facilitando a manutenção da consistência dos dados do paciente.

6.4.3 Modelação dos Dados dos Registos Clínicos

Cada Registo Clinico inclui dados detalhados de cada atendimento realizado, que inclui:

- Código de atendimento (`ID_Atendimento`)
- Código do paciente (`ID_Paciente`)
- Data e hora (`Data_Atendimento`)
- Informações sobe o responsável de saúde (`Responsavel`)
  - Código do responsável (`Responsavel.ID_Profisional`)
  - Nome do responsável (`Responsavel.Nome_Completo`)
  - Especialidade envolvida (`Responsavel.Especialidade`)
  - Experiência do responsável (`Responsavel.Anos_Experiencia`)
- Tipo do diagnóstico realizado pelo o responsável (`Tipo_Diagnostico`)
- Código CID10 como identificação do diagnóstico (`Codigo_CID10`)
- Curta descrição do diagnóstico realizado (`Descricao_Diagnostico`)
- Categorização do diagnóstico crónico (`Diagnostico_Cronico`)
- Histórico de Tratamentos prescritos pelo o responsável (`Tratamentos`)
- Informação do Paciente associado a este Registo Clinico (`Paciente`)

_id	ObjectID
ID_Atendimento	Number
ID_Paciente	Number
Data_Atendimento	Date
Responsavel	Object
Tipo_Diagnostico	String
Codigo_CID10	String
Descricao_Diagnostico	String
Diagnostico_Cronico	String
Tratamentos	Tratamentos[ ]
Paciente	Object

```
{
  _id: ObjectID
  ID_Atendimento: Number,
  ID_Paciente: Number,
  ID_Profissional: Number,
  Data_Atendimento: Date,

  // Responsavel de sáude
  Responsavel: {
     _id: ObjectID,
    ID_Profissional: Number,
    Nome_Completo: String,
    Especialidade: String,
    Anos_Experiencia: Number
  },

  Tipo_Diagnostico: String,
  Codigo_CID10: String,
  Descricao_Diagnostico: String,
  Diagnostico_Cronico: String,

  // Tratamentos prescritos pelo o Responsavel
  Tratamentos: [{ ... }],

  Paciente: {
        _id: ObjectID,
    ID_Paciente: Number,
    Nome_Completo: String,
    Data_Nascimento: Date,
    Idade: Number,
    Faixa_Etaria: String,
    Genero: String,
        Email: String,
        Telefone: String,
  },
}
```

Para os Registos Clínicos, foi implementada uma estratégia que combina referências externas com dados incorporados. As referências são utilizadas para relacionamentos com pacientes, profissionais, tratamentos, permitindo consultas independentes e facilitando a geração de relatórios estatísticos. Por outro lado, diagnósticos e tratamentos são incorporados diretamente no registo, mantendo o contexto completo do atendimento numa única unidade atómica. Esta abordagem permite uma escalabilidade horizontal eficiente e otimiza as consultas frequentes de histórico médico.

6.4.4 Modelação de Dados das Transferências

Cada `Registos de Transferencia` contém as seguintes informações:

- Codigo de Transferencia (`ID_Transferencia`)
- Codigo do paciente (`ID_Paciente`)
- Data da transferência (`Data_Transferencia`)
- Motivo (`Motivo`)
- Destino (`Destino`)
- Tipo de transferência (`Tipo_Transferencia`)
- Nome do Hospital de Origem (`Hospital_Origem`)
- Relatório médico associado, incluindo diagnósticos prévios e tratamentos realizados antes da transferência (`Relatório_Médico`)

_id	ObjectID
ID_Transferencia	Number
ID_Paciente	Number
ID_Profissional	Number
Data_Transferencia	Date
Motivo	String
Destino	String
Tipo_Transferencia	String
Hospital_Origem	String
Relatorio_Medico	Registo_Clinicos[ ]

```
{
  _id: ObjectID,
  ID_Transferencia: Number,
  ID_Paciente: Number,
  Data_Transferencia: Date,
  Motivo: String,
  Destino: String,
  Tipo_Transferencia: String,
  Hospital_Origem: String,

  // Relatorio médico associado incluindo vários RegistosClinicos
  Relatorio_Medico: [{ ... }]
}
```

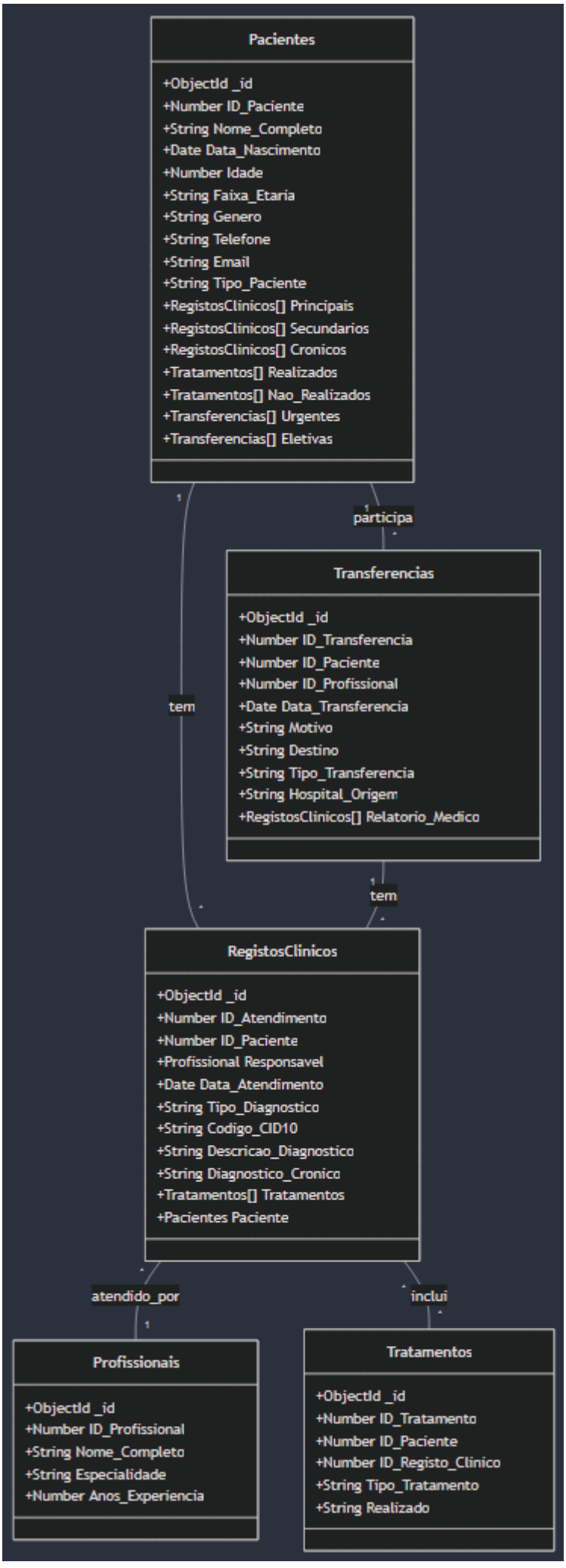
Na modelação das Transferências, usamos uma estrutura, utilizando referências para pacientes e dados incorporados para informações dos hospitais de origem e destino. Esta decisão reduz a necessidade de múltiplas consultas para obter detalhes completos de uma transferência, além de manter um histórico do paciente até ao momento da transferência. A estrutura facilita significativamente a geração de relatórios estatísticos e análises de fluxo de pacientes entre instituições.

6.5 Justificação e Resultado após Modelação

A modelação também considerou as necessidades específicas de relatórios mensais e análises estatísticas, um requisito fundamental do consórcio `MedSync`. A estrutura implementada foi especificamente desenhada para suportar a geração de relatórios. Ao nível operacional, permite extrair facilmente métricas como número de pacientes atendidos por faixa etária e género, quantidade de casos por especialidade médica, e volume de tratamentos realizados. Com este modelo é facilitado o acompanhamento da evolução temporal dos diagnósticos e dos tratamentos de cada paciente .

O design da base de dados foi otimizado para consultas agregadas através do framework de agregação do `MongoDB`, permitindo a geração eficiente de estatísticas complexas sem comprometer o desempenho do sistema.

Para suportar análises históricas mais profundas, o modelo também incorpora mecanismos de rastreamento temporal, mantendo o histórico completo de alterações em registos clínicos.


Pacientes
+ObjectId _id
+Number ID_Paciente
+String Nome_Completo
+Date Data_Nascimento
+Number Idade
+String Faixa_Etaria
+String Genero
+String Telefone
+String Email
+String Tipo_Paciente
+RegistosClinicos[] Principais
+RegistosClinicos[] Secundarios
+RegistosClinicos[] Cronicos
+Tratamentos[] Realizados
+Tratamentos[] Nao_Realizados
+Transferencias[] Urgentes
+Transferencias[] Eletivas
1
tem
1
participa
Transferencias
+ObjectId _id
+Number ID_Transferencia
+Number ID_Paciente
+Number ID_Profissional
+Date Data_Transferencia
+String Motivo
+String Destino
+String Tipo_Transferencia
+String Hospital_Origem
+RegistosClinicos[] Relatorio_Medico
1
tem
RegistosClinicos
+ObjectId _id
+Number ID_Atendimento
+Number ID_Paciente
+Profissional Responsavel
+Date Data_Atendimento
+String Tipo_Diagnostico
+String Codigo_CID10
+String Descricao_Diagnostico
+String Diagnostico_Cronico
+Tratamentos[] Tratamentos
+Pacientes Paciente
atendido_por
1
inclui
Profissionais
+ObjectId _id
+Number ID_Profissional
+String Nome_Completo
+String Especialidade
+Number Anos_Experiencia
Tratamentos
+ObjectId _id
+Number ID_Tratamento
+Number ID_Paciente
+Number ID_Registo_Clinico
+String Tipo_Tratamento
+String Realizado

7. Implementação

Esta secção descreve a implementação técnica do `MedSync`, abordando a **estrutura da API** a **integração com o** `MongoDB`, a **conversão dos dados para** `XML` utilizando XQuery e os **exemplos de consultas e transformações utilizadas**.

7.1 Estrutura de Dados e API

A API foi projetada para permitir o acesso aos dados clínicos armazenados no `MongoDB`. No enunciado refer que nos era permitido implementar uma REST API através do `Mongo Atlas` ou utilizando `Express JS` com `Node JS`, visto que a primeira está obsoleta e não foi capaz de realizar a mesma, optamos pela segunda opção, sendo desenvolvida da seguinte forma:

- `Node.js` e `Express.js` para gerir as rotas e as requisições `HTTP`.
- `MongoDB` como base de dados para armazenar os registos clínico e transferências .

Com isto a estrutura do projeto foi organizada da seguinte forma:

```
/TP_PEI_AC
|-- routes/                              # Pasta com as routes necessarias
|               |-- registosClinicos.js         # Rotas para registos
clinicos
|               |-- transferencias.js           # Rotas para transferencias
|-- .env                                 # MONGO_URI com conexao para o mongo
|-- index.js                             # Configuração principal da API
|-- package.`JSON`
|-- node_modules/
```

A API disponibiliza os seguintes endpoints para conexão com o `MongoDB`:

Método	Endpoint	Descrição
GET	`/api/registosClinicos?mes=XX&ano=YYYY`	Retorna os registos clínicos do mês.
GET	`/api/transferencias?mes=XX&ano=YYYY`	Retorna as transferências do mês.

7.2 Funcionamento dos Endpoints

A API REST permite a extração das informações dos registos clínicos e transferências (como dito anteriormente) . De seguida, detalhamos o funcionamento de cada endpoint:

7.2.1 API - Registos Clínicos

- **Método**: `GET`
- **Endpoint**: `/api/registosClinicos?mes=XX&ano=YYYY`
- **Descrição:** Permite obter **todos os registos clínicos** de um determinado mês/ano

Funcionamento

1. O cliente envia uma requisição com os parâmetros mês e ano.
2. A API valida os parâmetros e consulta o `MongoDB`, filtrando os dados com base nos
3. parâmetros que foram recebidos (mês/ano).
4. Os registos são retornados em formato `JSON` contendo as seguintes informações mostradas no screenshot a seguir:

```
▼ 0:
    _id:                    "677369dd5302a0db11bf983c"
    ID_Atendimento:         2
    ID_Paciente:            4030
    ID_Profissional:        29
    Data_Atendimento:       "2023-07-25T00:00:00.000Z"
  ▼ Responsavel:
      _id:                  "677307dbf5a217dbac62bdd8"
      ID_Profissional:      29
      Nome_Completo:        "Mackenzie Price"
      Especialidade:        "Ortopedista"
      Anos_Experiencia:     13
    Tipo_Diagnostico:       "Principal"
    Codigo_CID10:           "D50"
    Descricao_Diagnostico:  "Anemia por deficiência de ferro"
    Diagnostico_Cronico:    "Nao"
  ▼ Tratamentos:
    ▼ 0:
        _id:                "6772d57bd118ee0253ca8ed4"
        ID_Tratamento:      54897
        ID_Paciente:        4030
        ID_Registo_Clinico: 2
        Tipo_Tratamento:    "Endoscopia digestiva"
        Realizado:          "Sim"
    ▶ 1:                    {…}
    ▶ 2:                    {…}
  ▼ Paciente:
      ID_Paciente:          4030
      Nome_Completo:        "Joshua Robbins"
      Data_Nascimento:      "2004-07-27T00:00:00.000Z"
      Faixa_Etaria:         "Adulto"
      Género:               "F"
      Email:                "não fornecido"
```

7.2.2 API - Transferências Hospitalares

- **Método**: `GET`
- **Endpoint:** `/api/transferencias?mes=XX&ano=YYYY`
- **Descrição**: Obtém a lista de **transferências médicas** realizadas em um período específico.

Funcionamento:

1. O cliente envia uma requisição com os parâmetros `mês` e `ano`.
2. A API valida os parâmetros e consulta o `MongoDB`, filtrando os dados com base nos parâmetros que foram recebidos (mês/ano).
3. Os registos são retornados em formato `JSON` contendo as seguintes informações mostradas no screenshot a seguir:

```
▼ 0:
    _id:                    "677375805302a0db11ca486b"
    ID_Transferencia:       23
    ID_Paciente:            4222
    Data_Transferencia:     "2023-01-17T00:00:00.000Z"
    Motivo:                 "Consulta de segunda opinião"
    Destino:                "Centro de Reabilitação"
    Tipo_Transferencia:     "Urgente"
    Hospital_Origem:        "Hospital Geral da Cidade"
  ▼ Relatorio_Medico:
    ▼ 0:
        _id:                "677369f45302a0db11c088ad"
        ID_Atendimento:     24534
        ID_Paciente:        4222
        ID_Profissional:    1
        Data_Atendimento:   "2022-11-26T00:00:00.000Z"
      ▼ Responsavel:
          _id:              "677307dbf5a217dbac62bdbc"
          ID_Profissional:  1
          Nome_Completo:    "Trevor Donaldson"
          Especialidade:    "Dermatologista"
          Anos_Experiencia: 16
        Tipo_Diagnostico:   "Principal"
        Codigo_CID10:       "Q20"
        Descricao_Diagnostico: "Malformações congênitas do coração"
        Diagnostico_Cronico: "Sim"
      ▼ Tratamentos:
        ▼ 0:
            _id:            "6772d5b2d118ee0253cde95d"
            ID_Tratamento:  274682
            ID_Paciente:    4222
            ID_Registo_Clinico: 24534
            Tipo_Tratamento: "Consulta de saúde mental"
            Realizado:      "Sim"
    ▶ 1:                    {…}
    ▶ 2:                    { }
```

7.3 Integração com a Base de Dados MongoDB

A API está conectada à base de dados **HERE_WE_GO_AGAIN** no `MongoDB` onde todas as informações médicas estão armazenadas. A conexão com a base de dados é feita utilizando a biblioteca `MongoClient`, permitindo operações como:

- **Procura de registos clínicos** com base na data de atendimento.
- **Filtragem de transferências** por mês e ano.

A base de dados no `MongoDB` tem várias coleções (como ja foram referidas anteriormente no tópico modelação de dados) mas neste caso vamos só mostrar as que são usadas pela API:

Coleção	Descrição	
`RegistosClinicosAPI`	Armazena os dados dos atendimentos, incluindo informações do paciente, diagnósticos, tratamentos e informação do responsável	
`TransferenciasAPI`	Armazena as informações de transferências como id da transferência, data, relatório medico, etc..	

7.4 Fluxo de Requisição e Resposta

O fluxo do funcionamento da API pode ser descrito da seguinte forma:

1. **Requisição:**
   - O utilizador faz uma requisição para a API, especificando o mês e o ano desejado.
2. **Validação:**
   - A API verifica se os parâmetros recebidos são válidos.
3. **Consulta ao `MongoDB`:**
   - Os registos clínicos/transferências são filtrados consoante os parâmetros recebidos.
4. **Retorna da resposta:**
   - A API devolve os dados encontrados em **formato `JSON`**, ou retorna uma mensagem de erro se não existir nenhum registo/transferência nesse período.

7.5 Explicação das transformações XQuery desenvolvidas

Após a obtenção dos registos clínicos e das transferências no **formato `JSON`** via API REST, foi necessário **converter esses dados para `XML`**, garantindo a conformidade com o padrão exigido pelo **MedSync**.

Optámos por utilizar **XQuery no BaseX** para a conversão, garantindo que os dados pudessem ser **estruturados e validados** conforme necessário

---

7.5.1 Conversão para XML

A conversão dos dados para `XML` é essencial por vários motivos:

- Padrão utilizado em sistemas hospitalares -> Muitas instituições utilizam `XML` para armazenar e trocar dados médicos, por exemplo o SNS utiliza uma API .
- Interoperabilidade -> `XML` facilita a comunicação entre diferentes softwares médicos.
- Organização Estruturada -> Permite armazenar registos de forma hierárquica, facilitando consultas

No nosso caso, os dados são recebidos pela API REST em `JSON` e depois convertidos para `XML`. O objetivo é gerar relatórios mensais de **registos clínicos** e **transferências**, estruturados de acordo com as especificações exigidas.

7.5.2 Estrutura do XML gerado

O `XML` que é gerado segue um estrutura padronizada (seguindo o que o enunciado pede) para representar os relatórios mensais dos registos clínicos e dos relatórios mensais das transferências.

Exemplo relatório registos clínicos `XML`:

```
−<RegistosMensais>
  −<HospitalInfo>
      <CodigoHospital>1234</CodigoHospital>
      <Nome>Hospital Ruben Pedro</Nome>
      <Morada>Rua do hospital, 1234</Morada>
      <DataRelatorio>07 - 2023</DataRelatorio>
    </HospitalInfo>
  −<PacientesPorFaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>11516</TotalPacientes>
        <FaixaEtaria>Adulto</FaixaEtaria>
        <Genero>F</Genero>
      </FaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>11310</TotalPacientes>
        <FaixaEtaria>Adulto</FaixaEtaria>
        <Genero>M</Genero>
      </FaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>2491</TotalPacientes>
        <FaixaEtaria>Idoso</FaixaEtaria>
        <Genero>F</Genero>
      </FaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>2410</TotalPacientes>
        <FaixaEtaria>Idoso</FaixaEtaria>
        <Genero>M</Genero>
      </FaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>931</TotalPacientes>
        <FaixaEtaria>Adolescente</FaixaEtaria>
        <Genero>M</Genero>
      </FaixaEtariaGenero>
    −<FaixaEtariaGenero>
        <TotalPacientes>994</TotalPacientes>
        <FaixaEtaria>Adolescente</FaixaEtaria>
        <Genero>F</Genero>
      </FaixaEtariaGenero>
    </PacientesPorFaixaEtariaGenero>
  −<CasosPorEspecialidade>
    −<Especialidade>
        <Nome>Ortopedista</Nome>
        <Casos>5991</Casos>
      </Especialidade>
    −<Especialidade>
        <Nome>Clinico Geral</Nome>
        <Casos>6262</Casos>
      </Especialidade>
    −<Especialidade>
        <Nome>Dermatologista</Nome>
        <Casos>7106</Casos>
      </Especialidade>
    −<Especialidade>
        <Nome>Pediatra</Nome>
        <Casos>4442</Casos>
      </Especialidade>
    −<Especialidade>
        <Nome>Cardiologista</Nome>
        <Casos>5851</Casos>
      </Especialidade>
    </CasosPorEspecialidade>
    <NumeroTratamentosRealizados>47976</NumeroTratamentosRealizados>
    <NumeroCondicoesCronicasDiagnosticadas>8697</NumeroCondicoesCronicasDiagnosticadas>
  </RegistosMensais>
```

Exemplo relatório transferências mensais:


```
−<RelatorioTransferenciasMensais>
    <TotalTransferencias>921</TotalTransferencias>
  −<TransferenciasPorMotivo>
    −<Motivo>
        <Descricao>Consulta de segunda opinião</Descricao>
        <Total>180</Total>
      </Motivo>
    −<Motivo>
        <Descricao>Tratamento especializado</Descricao>
        <Total>157</Total>
      </Motivo>
    −<Motivo>
        <Descricao>Reabilitação prolongada</Descricao>
        <Total>136</Total>
      </Motivo>
    −<Motivo>
        <Descricao>Procedimento cirúrgico avançado</Descricao>
        <Total>142</Total>
      </Motivo>
    −<Motivo>
        <Descricao>Urgência médica</Descricao>
        <Total>146</Total>
      </Motivo>
    −<Motivo>
        <Descricao>Exames complementares de diagnóstico</Descricao>
        <Total>160</Total>
      </Motivo>
  </TransferenciasPorMotivo>
  −<TransferenciasPorTipo>
    −<Tipo>
        <Descricao>Urgente</Descricao>
        <Total>473</Total>
      </Tipo>
    −<Tipo>
        <Descricao>Eletiva</Descricao>
        <Total>448</Total>
      </Tipo>
  </TransferenciasPorTipo>
  −<TransferenciasPorHospital>
    −<Hospital>
        <Nome>Hospital Geral da Cidade</Nome>
        <Total>161</Total>
      </Hospital>
    −<Hospital>
        <Nome>Hospital Santa Maria</Nome>
        <Total>157</Total>
      </Hospital>
    −<Hospital>
        <Nome>Hospital de Coração</Nome>
        <Total>149</Total>
      </Hospital>
    −<Hospital>
        <Nome>Hospital São João</Nome>
        <Total>150</Total>
      </Hospital>
    −<Hospital>
        <Nome>Clínica São Paulo</Nome>
        <Total>156</Total>
      </Hospital>
    −<Hospital>
        <Nome>Centro Hospitalar do Norte</Nome>
        <Total>148</Total>
      </Hospital>
  </TransferenciasPorHospital>
</RelatorioTransferenciasMensais>
```

7.5.3 Processo de conversão

A conversão do `JSON` para `XML` é feita utilizando o `XQuery` usado no `BaseX` , este processo envolveu **três etapas principais**:

1. **Extração dos dados a API REST ->** A API recebe as requisições e retorna os dados em formato `JSON`.
2. **Transformação dos dados para `XML` ->** O `XQuery` processa os dados `JSON` e converte-os num documento `XML` estruturado.
3. **Geração do relatório final ->** O `XML` gerado é formatado de acordo com os requisitos do enunciado para garantir a compatibilidade entre hospitais.

7.5.4 Detalhes sobre a integração entre diferentes componentes

A integração entre os componentes foi projetada para garantir a comunicação eficiente entre a **API REST**, a base de dos `MongoDB` e o BaseX para as transformações `XQuery`. A API REST foi criada com `Node.js` e `Epress.js`, enquanto os dados são armazenados no `MongoDB`. O `BaseX` é responsável pela transformação dos dados recebidos para `XML`.

7.5.5 Fluxo da Integração

1. **Requisição para a API REST:**
   - O usuário faz uma requisição para a API, especificando o mês e o ano desejados.
   - A API consulta o `MongoDB` para buscar os registos clínicos ou transferências.
2. **Consulta ao `MongoDB`:**
   - A API filtra os dados de acordo com o mês e ano fornecido pelo usuário, acedendo ás coleções apropriadas no `MongoDB`.

3. **Transformação para `XML` no `BaseX`:**
    - Após obter os dados, a API converte as informações em `JSON` para `XML` utilizando `BaseX` e `XQuery`.
    - O `XML` gerado é estruturado conforme os requisitos do `MedSync`.
4. **Geração do Relatório:**

- O relatório final em `XML` é enviado para o usuário, pronto para ser integrado com outros sistemas hospitalares.

Essa abordagem garante a **interoperabilidade** e a **padronização** dos dados, facilitando a troca de informações entre os sistemas de saúde e proporcionando a geração de relatórios conforme as exigências do enunciado.

7.6 Justificação da abordagem escolhida

A abordagem utilizada no `MedSync` foi escolhida com base em **eficiência, escalabilidade e compatibilidade com padrões hospitalares**.

Decisão	Motivo
`Node.js` + `Express.js`	Flexibilidade, melhor controlo da API e suporte a `JSON`.
`MongoDB`	Modelo flexível para registos clínicos, consultas rápidas e integração nativa com o `JSON`
`BaseX` + `XQuery` para `XML`	Melhor interoperabilidade, suporte para `XML` estruturado e compatibilidade com sistemas hospitalares.

Dessa forma, a arquitetura do `MedSync` assegura que os dados clínicos possam ser **armazenados, extraídos e compartilhados de forma eficiente**, garantindo **a interoperabilidade entre hospitais**.

8. Apreciação Critica

Esta secção apresenta uma análise dos desafios encontrados durante o desenvolvimento do projeto `MedSync`, a justificativa das decisões técnicas adotadas, as limitações identificadas na solução atual e possíveis melhorias para futuras versões do sistema.

8.1 Análise dos Desafios Encontrados

O desenvolvimento do `MedSync` apresentou **vários desafios técnicos e estruturais**, que exigiram um estudo aprofundado e a tomada de decisões estratégicas. Os principais desafios forma:

1. **Estruturação e Modelação dos Dados no `MongoDB`**
   - Os ficheiros `csv` fornecidos não seguiam um **modelo de dados unificado**, o que exigiu da nossa parte uma estruturação melhor desses dados, no início tivemos dificuldade pois haviam ficheiros que não tinham uma relações, sendo nós obrigados a "editar esses mesmos".
   - Foi necessário definir uma **indexação** para otimizar as consultas em grandes volumes de dados, mesmo assim a otimização não era muito rápida.
2. **Conversão de `JSON` para `XML` via `XQuery`**
   - Para que fosse possível a conversão dos dados `JSON` para `XML` e que estes mesmos seguissem o padrão pedido no enunciado, foi preciso organizar previamente os dados enviados para a API REST, para que depois estes mesmo fossem convertidos em `XML` com a estrutura pedida.
3. **Otimização a API REST**
   - Inicialmente, a API REST apresentava **tempo de resposta elevado** ao processar grandes volumes de dados, pois estávamos a fazer `aggregates, unwinds e projects` diretamente na API o que resultava um tempo bastante grande de consulta.

- Foi necessário otimizar os dados ainda mais (no `MongoDB`) para que a API REST fizesse só um `find` com as datas pedidas pelo utilizador

4. **Integração Entre Componentes**
   - Garantir que a API REST, o `MongoDB` e o `BaseX` funcionassem **de forma integrada e eficiente** foi um desafio.
   - Sem sabermos explicar o porquê descobrimos que certos campos, ao criar a `XQuery` no `BaseX`, deveriam de levar dois "__" e não um como estava no `JSON` original
   - Algumas bibliotecas para a conversão de `JSON` para `XML` apresentavam **limitações**, o que levou à decisão de usar `XQuery` **diretamente** no `BaseX`.
5. **Gestão de Dados Omissos**
   - Alguns ficheiros `csv` não tinham qualquer relação entre eles, como ja foi referido anteriormente, o ficheiro `Tratamentos.csv` não tinha nenhum relação com os outros ficheiros presentes na base de dados, o que nos dificultou no início estabelecer uma relação entre eles, pois estes mesmo eram necessários e imprescindíveis para o trabalho.

8.2 Discussão das Decisões Técnicas Tomadas

Várias decisões técnicas foram tomadas ao longo do desenvolvimento do `MedSync`, com base em fatores como **desempenho, escalabilidade e compatibilidade**.

1. **Uso de `Node.js` + `Express.js` para a API REST**
   - No enunciado foi nos proposto que poderíamos usar tanto o `Mongo Atlas` ou o Node.js, seguimos como estava nos slides das aulas práticas e não encontramos, no `Mongo Atlas`, a parte da API, então resolvemos fazer em `Node.js` + `Express.js`.
   - O Node.js é **leve** e **eficiente** para aplicações baseadas em `JSON`.
   - `Express.js` permite criar endpoints REST de forma modular e escalável.
   - Maior **flexibilidade e controlo** em comparação com o `Mongo Atlas`, que tem limitações.
2. **Estratégias de Modelação no `MongoDB`**
   - **Document Embedding:** Para melhorar a **performance** de consultas em registos clínicos, tratamentos foram embutidos diretamente nos documentos dos registos clínicos.

- Como queríamos que a API recebesse as informações todas num `JSON` só (não houvesse a necessidade de dizer a API para fazer `aggregates` etc..), relacionamos todas as informações necessárias numa so coleção.

8.3 Identificação de Limitações da Solução

Apesar da solução implementada ser funcional e eficiente, ainda existem algumas **limitações** que podem ser aprimoradas em versões futuras:

1. **Falta de Autenticação na API REST**
   - Atualmente, a API REST **não implementa mecanismos de autenticação e autorização**, o que pode ser um risco em um ambiente hospitalar real.
2. **Ausência de Interface Gráfica para Visualização dos Relatórios**
   - Os relatórios gerados estão disponíveis apenas em **formato** `XML`, o que pode dificultar a leitura para utilizadores não técnicos.
3. **Dependência do `BaseX` para Conversão `XML`**
   - Atualmente, a conversão para `XML` só é possível via `BaseX`, o que pode limitar a portabilidade do sistema.
4. **Otimização de Performance para Grandes Volumes de Dados**
   - Com o crescimento do volume de registos, as consultas no `MongoDB` podem tornar-se mais lentas.
5. **Atualização de Dados em Tempo Real**
   - A API funciona atualmente como um sistema de **consulta estática**, sem atualização automática dos dados.

8.4 Sugestões de Melhorias Futuras

Com base nas limitações identificadas, propomos as seguintes melhorias para versões futuras do `MedSync`:

Melhoria	Descrição
Autenticação na API REST	Implementar `JWT` ou `OAuth` para garantir segurança no acesso aos dados.

Melhoria	Descrição
Dashboard Web	Criar uma interface gráfica para facilitar a visualização dos relatórios `XML` .
Conversão `JSON` → `XML` na API	Reduzir a dependência do BaseX implementando conversão `XML` diretamente na API.

9. Conclusão

Com a implementação deste trabalho ficou claro que a interoperabilidade dos dados são importantes, principalmente em ambientes hospitalares, garantindo assim um fluxo eficiente e seguro das informações entre diferentes hospitais. Ao longo do projeto, foram aplicadas várias técnicas de modelação dos dados (pois estes não vinham estruturados da melhor forma) para que fosse possível otimizar as consultas e a integração nos sistemas dos mesmo, resultando numa solução funcional e escalável para a partilha de registos clínicos e transferências entre hospitais.

9.1 Síntese do Trabalho Realizado

O projeto focou-se na construção de um sistema capaz de **extrair, processar e estruturar dados clínicos**. As principais etapas desenvolvidas foram:

- **Extração de Dados:** Conversão dos ficheiros `csv` para `MongoDB`, garantindo uma estruturação otimizada das informações.
- **Modelação da Base de Dados:** Organização eficiente das coleções no `MongoDB`, garantindo consultas rápidas.
- **Desenvolvimento da API REST:** Implementação de endpoints em `Node.js` e **Express.js** permitindo a consulta eficiente dos registos clínicos e transferências.
- **Conversão JSO->`XML`:** Utilização do `XQuery` e `BaseX` para transformar os dados da API num formato padronizado.
- **Geração de Relatórios**: Desenvolvido no `XQuery` a geração dos relatórios mensais em formato `XML`.

9.2 Avaliação dos Objetivos Alcançados

Os objetivos definidos para o projeto foram amplamente alcançados, garantindo um **sistema funcional e otimizado** para a gestão de dados clínicos. Abaixo, apresentamos a avaliação de cada objetivo estabelecido:

Objetivo	Comentário
Criar um modelo eficiente de armazenamento de dados clínicos	O `MongoDB` foi utilizado com padrões modernos de modelação.
Implementar uma API REST para consulta e integração dos dados	API construída com Node.js e Express.js, com endpoints bem definidos.
Garantir a conversão de `JSON` para `XML` para compatibilidade hospitalar	Utilização de XQuery e BaseX para gerar `XML` estruturado.
Otimizar o desempenho do sistema para consultas rápidas	Indexação e modelação eficiente reduziram os tempos de resposta das queries.
Garantir a interoperabilidade entre diferentes hospitais	O formato `XML` gerado pode ser facilmente integrado com outros sistemas de saúde.

9.3 Reflexão sobre as Aprendizagens Adquiridas

O desenvolvimento do trabalho proporcionou-nos diversas aprendizagens técnicas, tanto no domínio de base de dados no `MongoDB`, como na implementação de uma API REST utilizando o `Node.js` com `Express.js`. Algumas das principais aprendizagens incluem:

1. **Domínio no `MongoDB` e Modelação de Dados**
    - A estruturação eficiente das coleções diminui o tempo de consulta e fazer so coleções para apresentar os dados pretendidos.
    - O uso de padrões como **Document Embedding** e **Computed Pattern** melhorou o desempenho da Anos_Experiencia
    - Muito mais pro eficientes a trabalhar com o `MongoDB` e as suas funcionalidades

2. **Implementação de APIs REST**
    - Como nós nunca tínhamos feito uma API REST muito menos usando o Express.js e Node.js, foi uma novidade para nós ao qual enfrentamos o desafio e aprendemos bastante com o mesmo.
    - O **Express.js** e o `MongoDB` proporcionaram um desenvolvimento ágil e eficiente da API.
3. **Transformação de Dados `JSON` -> `XML` com `XQuery`**
    - A conversão para `XML` foi um desafio ao qual nos adquirimos ainda mais conhecimentos sobre `XQuery` e `BaseX`.
    - O uso de `XQuery` permitiu estruturar os dados de forma flexível e adaptável.
4. **Gestão de Desafios e Resolução de Problemas**
    - Aprendemos a **lidar com dados omissos e inconsistentes**.
    - Tivemos de ver quais dados estavam omissos, e resolver esse problema.
    - Adaptámos a modelação dos dados conforme as **necessidades do projeto** e dos **hospitais envolvidos**.

---

10. Conclusão Geral

O desenvolvimento do trabalho permitiu a construção de um modelo robusto e otimizado para consulta de registos clínicos e transferências, abordando as complexidades inerentes à manipulação de grandes volumes de dados.

A modelação dos dados foi estruturada de forma a equilibrar a flexibilidade e o desempenho, permitindo consultas rápidas. A implementação da API REST facilitou a recuperação e manipulação das informações, promovendo um acesso dinâmico e parametrizado aos dados clínicos e às transferências. Além disso, a utilização de `XQuery` e `BaseX` para a conversão `JSON`->`XML` demonstra-se principalmente para assegurar a compatibilidade entre os hospitais.

O projeto alcançou os objetivos propostos, validando a eficácia da abordagem modular na modelação dos dados e destacando a importância da estruturação eficiente de base de dados. Através de uma analise critica, foi possível desenvolver um sistema otimizado, mitigando desafios como dados omissos, necessidades de consultas escaláveis e até mesmo a densidade de dados.

Como perspectiva futura(como ja referido no relatório), o trabalho pode ser expandido com **novos mecanismos de segurança e dashboards de monitorização em tempo real**, consolidando-se como uma solução viável para a modernização da infraestrutura digital dos serviços de saúde.