

Disciplina de Processamento Estruturado de Informação

Ano Letivo de 2024/2025

MedSync

8200489 | João Pedro Teixeira

8220360 | Rafael Saraiva

8210666 | Manuel Pereira

Felgueiras, janeiro de 2025

Resumo

Este trabalho foi realizado para desenvolver uma solução que integre e padronize a partilha de dados médicos entre hospitais MedSync, promovendo a continuidade e a qualidade dos cuidados de saúde. O projeto foi estruturado em três etapas: extração de dados médicos em formato CSV e organização em MongoDB, disponibilização desses dados via API REST em formato JSON e transformação para documentos XML padronizados utilizando XQuery no BaseX. Os principais resultados incluem uma estrutura de dados eficiente, a geração de relatórios clínicos e de transferências em conformidade com o vocabulário XML definido e a validação do mesmo recorrendo a XSD.

Índice

Resumo	2
Introdução	5
Motivação e Objetivos	6
Apresentação e discussão dos resultados	7
MongoDB.....	7
Transformação e organização dos dados	7
Pipelines	8
API REST	11
BASEX.....	14
XSD	16
POSTMAN	17
Conclusão.....	18

Índice de Ilustrações

Figura 1 - Estrutura da coleção transfers_by_date	8
Figura 2 - Estrutura da coleção records_by_date	9
Figura 3 - Conexão à base de dados no servidor em node.js	11
Figura 4 - Controlador das transferências (node.js)	12
Figura 5 - Pedido HTTP GET ao servidor para obter os registos Clínicos	13
Figura 6 - Pedido HTTP GET ao servidor para obter as transferências	13
Figura 7 - XQuery usado para os registos clínicos	14
Figura 8 - Resposta do BaseX com os Registos Clínicos padronizado	15
Figura 9 - Resposta do BaseX com o relatório de transferências padronizado	15
Figura 10 - extrato simples do .xsd	16
Figura 11 - Extrato da estrutura do ficheiro .xsd	16
Figura 12 - Interface do Postman	17

Introdução

A partilha eficiente de dados médicos é fundamental para garantir a qualidade dos cuidados de saúde, especialmente em cenários que envolvem múltiplos hospitais. O projeto MedSync surge como uma resposta à necessidade de integração de informações clínicas entre instituições parceiras, promovendo uma abordagem padronizada para a gestão e troca de dados.

Este relatório apresenta o desenvolvimento de um sistema que suporta a extração, transformação e disponibilização de dados médicos. No âmbito deste trabalho, foi implementado um processo composto pelas seguintes etapas principais: a integração e transformação dos dados em MongoDB, o desenvolvimento de uma API REST em node.js, conversão dos dados em formato JSON para um formato XML padronizado utilizando XQuery e validando os mesmos com um XSD.

Além de contextualizar o caso de estudo, este documento descreve a modelação dos dados em MongoDB, as consultas implementadas para extração de informações relevantes, a criação da API REST e os scripts de transformação XML.

Motivação e Objetivos

O principal objetivo deste projeto é desenvolver um sistema que suporte o fluxo de dados médicos entre hospitais, desde a extração até à apresentação padronizada dos mesmos.

Pretendemos atingir este objetivo por meio das seguintes etapas:

1. **Extração e Organização dos Dados:** Importar dados médicos disponibilizados em formato CSV para uma base de dados MongoDB, garantindo uma estrutura consistente que atenda aos requisitos definidos.
2. **Disponibilização via API REST:** Implementar uma API REST para permitir o acesso aos dados integrados em formato JSON, facilitando a consulta e manipulação das informações.
3. **Transformação para XML:** Utilizar scripts XQuery no BaseX para converter os dados JSON em documentos XML padronizados, de acordo com o vocabulário definido pelo MedSync.

As ferramentas escolhidas para alcançar esses objetivos são:

- **MongoDB:** Para armazenar e estruturar os dados devido à sua flexibilidade e capacidade de lidar com documentos heterogêneos.
- **Express.js e Node.js:** Para desenvolver a API REST, oferecendo uma interface eficiente e acessível para os dados.
- **BaseX:** Para realizar a transformação dos dados para XML utilizando XQuery, assegurando a conformidade com o vocabulário exigido.
- **Oxygen XML Editor:** Para a criação dos ficheiros .xsd, de forma a validar os documentos XML gerados pelas consultas

Com estas etapas e ferramentas, pretendemos criar uma solução que permita a integração eficiente dos dados médicos

Apresentação e discussão dos resultados

MongoDB

O **MongoDB** é uma base de dados de natureza orientada a documentos, que oferece flexibilidade e facilidade de integração de diferentes tipos de dados. Este modelo permite armazenar informações heterogéneas de maneira organizada e eficiente, características essenciais para atender à complexidade dos dados médicos tratados neste projeto.

Transformação e organização dos dados

Para atender às necessidades específicas deste projeto e facilitar o processo de gerar relatórios padronizados, foram criadas duas novas coleções chamadas ***transfers_by_date*** e ***records_by_date***.

Transfers_by_date - consolida informações de transferências médicas, registos clínicos e tratamentos prévios, estruturando os dados de forma otimizada para consultas futuras, organizando por ano e mês.

Records_by_date - consolida informações consolidar informações de diagnósticos e tratamentos, organizando por ano e mês.

Pipelines

Na pipeline que transformou os documentos de transferências em documentos otimizados para este projeto foram utilizadas as seguintes estratégias, respetivamente:

1. **\$match**: Filtra os documentos para incluir apenas aqueles em que a Data_Atendimento no clinical_records é anterior à Data_Transferencia.
2. **\$match**: Filtra os documentos para incluir aqueles onde a Data_Tratamento é anterior à Data_Transferencia.
3. **\$group**: Agrupa os documentos por _id e coleta informações relevantes, como Destino, Motivo, e os diagnósticos e tratamentos anteriores.
4. **\$addFields**: Adiciona os campos Ano_Transferencia e Mes_Transferencia extraídos da Data_Transferencia.
5. **\$group**: Agrupa os documentos por ano criando uma lista de transferências para cada mês dentro de cada ano, e depois, agrupa novamente, agora por ano, criando uma estrutura hierárquica com meses e transferências.
6. **\$sort**: Ordena os documentos agrupados por ano de forma crescente.

Após a execução desta pipeline, a nova coleção é criada com a seguinte estrutura:

```
{
  "_id": 2023,
  "Meses": [
    {
      "Mes": 1,
      "Detalhes": [
        {
          "Destino": "Centro Cardiológico",
          "ID_Transferencia": 1638,
          "ID_Paciente": 13348,
          "ID_Profissional": 159,
          "Data_Transferencia": {
            "$date": "2023-01-15T00:00:00.000Z"
          },
          "Motivo": "Exames complementares de diagnóstico",
          "Tipo_Transferencia": "Urgente",
          "Diagnosticos_Previos": [
            {
              "Tipo_Diagnostico": "Principal",
              "Codigo_CID10": "O80",
              "Descricao_Diagnostico": "Parto único espontâneo"
            }
          ],
          "Tratamentos_Previos": [
            {
              "ID_Tratamento": 293849,
              "Tipo_Tratamento": "Losartana 50mg",
              "Realizado": "Sim"
            }
          ]
        }
      ]
    }
  ],
  "Ano": 2023
}
```

Figura 1 – Estrutura da coleção transfers_by_date

Na pipeline que transformou os documentos de registos clínicos em documentos otimizados para este projeto foram utilizadas as seguintes estratégias, respetivamente:

1. **\$group**: Agrupa os documentos por ID_Atendimento e coleta informações de diagnósticos, bem como os dados do paciente e profissional.
2. **\$addFields (Tratamentos)**: Combina os tratamentos e as atualizações de tratamentos numa única lista.
3. **\$addFields (Ano_Atendimento, Mes_Atendimento)**: Adiciona campos para o ano e mês extraídos da Data_Atendimento.
4. **\$group**: Agrupa os documentos por ano e mês (Ano_Atendimento e Mes_Atendimento), criando uma lista de registos para cada mês, e depois, agrupa os documentos por ano.
5. **\$sort**: Ordena os documentos por ano de forma crescente.

Após a execução desta pipeline, a nova coleção é criada com a seguinte estrutura:

```
{
  "_id": 2022,
  "Meses": [
    {
      "Mes": 12,
      "Registos": [
        {
          "ID_Registo_Clinico": 31,
          "ID_Paciente": 4853,
          "ID_Profissional": 67,
          "Data_Atendimento": {
            "$date": "2022-12-24T00:00:00.000Z"
          },
          "Diagnosticos": [
            {
              "Tipo_Diagnostico": "Principal",
              "Codigo_CID10": "N39",
              "Descricao_Diagnostico": "Outros distúrbios do trato urinário"
            }
          ],
          "Tratamentos": [
            {
              "_id": {
                "$oid": "676724464bf1a1b510c022e4"
              },
              "ID_Tratamento": 174779,
              "ID_Registo_Clinico": 31,
              "Tipo_Tratamento": "Aplicação de insulina",
              "Realizado": "Sim"
            }
          ]
        }
      ],
      "Pacientes": [
        {
          "_id": {
            "$oid": "676723564bf1a1b510bce63a"
          },
          "ID_Paciente": 4853,
          "Nome_Completo": "Eric Wilson",
          "Data_Nascimento": {
            "$date": "1984-05-09T00:00:00.000Z"
          },
          "Género": "M",
          "Email": "desconhecido",
          "Data_Registo": {
            "$date": "2019-11-26T00:00:00.000Z"
          }
        }
      ]
    }
  ],
  "Ano": 2022
}
```

Figura 2 – Estrutura da coleção records_by_date

Mais tarde, devido à necessidade de obter estatísticas mensais referentes ao mês que o utilizador pede, foram criadas duas pipeline capaz de filtrar pelo mês pedido e gerar estatísticas com base no conteúdo da coleção.

No seguinte ponto abordamos o essencial da pipeline que devolve os registos médicos de um mês.

- **\$addFields (estatísticas):** Adiciona um novo campo estatísticas, que contém:
 - **faixaEtaria:** Calcula a distribuição de pacientes por faixa etária, usando a idade calculada com base no ano de nascimento (Data_Nascimento). A faixa etária é dividida em 3 categorias: 0-18, 19-65 e 65+. O **\$reduce** percorre os pacientes e conta quantos se encaixam em cada faixa etária.
 - **porGenero:** Calcula a distribuição de pacientes por género (M para masculino e F para feminino), utilizando o campo Género de cada paciente.
 - **totalTratamentos:** Conta o total de tratamentos realizados durante o mês, somando o número de tratamentos (Tratamentos) presentes no campo Registos.

Nos dois seguintes pontos abordamos o essencial da pipeline que devolve as transferências de um mês.

- **\$group:** Reagrupa as transferências e calcula o total de transferências, além de criar dois conjuntos:
 - **transferenciasPorMotivo:** Conjunto de motivos distintos das transferências.
 - **transferenciasPorTipo:** Conjunto de tipos distintos de transferências.
- **\$project:** Formata o documento final, incluindo:
 - **totalTransferencias:** Total de transferências no mês.
 - **transferencias:** A lista completa de transferências.
 - **porMotivo:** Para cada motivo distinto de transferência, conta quantas transferências possuem esse motivo.
 - **porTipo:** Para cada tipo distinto de transferência, conta quantas transferências possuem esse tipo.

API REST

Devido à descontinuação da funcionalidade de DATA API do MongoDB sentimos a necessidade de criar uma API para solucionar este problema. A API REST baseia-se em dois métodos que servem de ligação entre a Base de Dados MongoDB e o BaseX. Esta API faz o papel de servidor sendo que em cada método realiza um pedido http ao MongoDB para obter os dados em formato JSON.

Foi necessário estabelecer uma ligação a base de dados e a API, para a obter as informações corretas fornecidas pelos hospitais. A figura seguinte mostra como foi estabelecida a ligação.

```
const { MongoClient } = require('mongodb');

const uri =
'mongodb+srv://pedro:pedro@medsync.vziha.mongodb.net/?retryWrites=true&w=majority&appName=MedSync';
const dbName = 'data';
let db;

async function connectToDb() {
  if (!db) {
    const client = new MongoClient(uri);
    await client.connect();
    console.log('Connected to MongoDB');
    db = client.db(dbName);
  }
  return db;
}

module.exports = connectToDb;
```

Figura 3 - Conexão à base de dados no servidor em node.js

A API tem dois controladores simples com apenas um método que serve para realizar os pedidos de informação à base de dados. Esta informação é devolvida em formato .json e posteriormente desserializada no BaseX.

A figura seguinte mostra um dos controladores criados.

```
const connectToDb = require('../db');

exports.getTransferReportByMonth = async (req, res) => {
  const {year, month} = req.query;

  if (!year || !month) {
    return res.status(400).json({success: false, message: "Year and month are required."});
  }

  const yearInt = parseInt(year, 10);
  const monthInt = parseInt(month, 10);

  try {
    const db = await connectToDb();
    const transfersCollection = db.collection('transfers_by_date');

    const pipeline = [...];

    const result = await transfersCollection.aggregate(pipeline).toArray();

    res.status(200).json({
      success: true,
      data: result,
    });
  } catch (error) {
    console.error(error);
    res.status(500).json({
      success: false,
      message: 'An error occurred while generating the report.',
    });
  }
};
```

Figura 4 – Controlador das transferências (node.js)

As seguintes figuras mostram como são devolvidos os dados a partir da API.

GET ▼ http://localhost:3000/api/clinical/report?year=2023&month=10 Try ↗

Params ● Headers Body

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	year	2023			
<input checked="" type="checkbox"/>	month	10			

Body Headers (7) Status Code 200 OK

Pretty Raw Preview JSON ▼ ≡

```
1 {
2   "success": true,
3   "data": [
4     {
5       "Ano": 2023,
6       "Mes": 10,
7       "Registos": [...],
741    },
742    "Pacientes": [...],
970    ],
971    "estatisticas": {
972      "faixaEtaria": {
973        "0-18": 3,
974        "19-65": 19,
975        "65+": 2
976      },
977      "porGenero": {
978        "M": 17,
979        "F": 7
980      },
981      "totalTratamentos": 27
982    }
983  ]
984 }
985 }
```

Figura 5 – Pedido HTTP GET ao servidor para obter os registos Clínicos

GET ▼ http://localhost:3000/api/transfers/report?year=2023&month=11 Try ↗

Params ● Headers Body

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	year	2023			
<input checked="" type="checkbox"/>	month	11			

Body Headers (7) Status Code 200 OK

Pretty Raw Preview JSON ▼ ≡

```
5   "totalTransferencias": 86,
6   "transferencias": [...],
15590  ],
15591  "porMotivo": [
15592    {
15593      "motivo": "Tratamento especializado",
15594      "totalPorMotivo": 14
15595    },
15596    { ... },
15599    { ... },
15600    { ... },
15603    { ... },
15604    { ... },
15607    { ... },
15608    { ... },
15611    { ... },
15612    { ... },
15615    { ... },
15616  ],
15617  "porTipo": [
15618    {
15619      "tipo": "Eletiva",
15620      "totalPorTipo": 43
15621    },
15622    { ... },
15625    { ... }
15626  ]
```

Figura 6 – Pedido HTTP GET ao servidor para obter as transferências

A informação devolvida é usada posteriormente pelo BaseX.

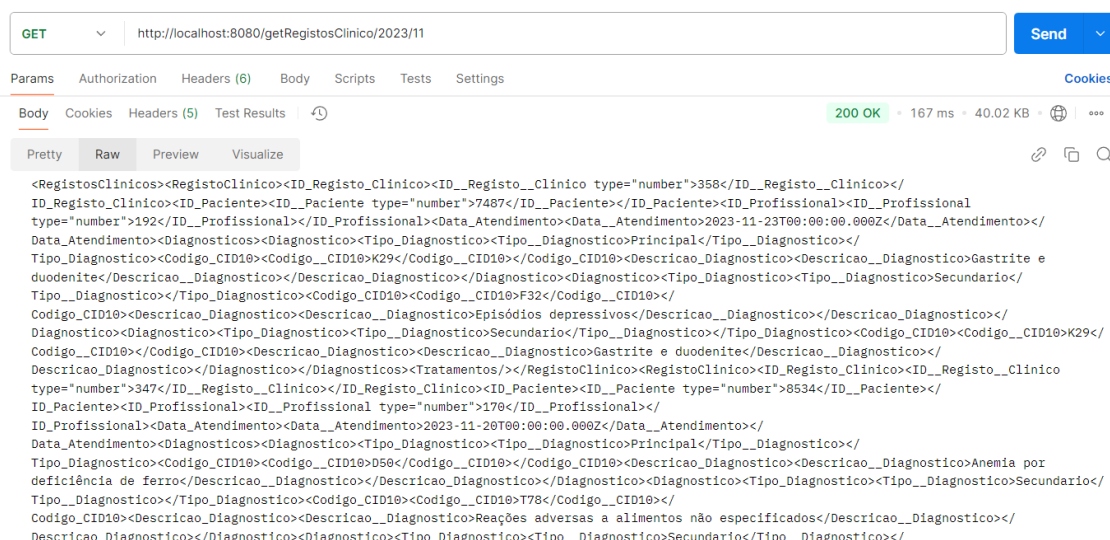
BASEX

O papel do BaseX baseia-se a manipulação dos dados, onde estes são entregues no formato .xml padronizado com a utilização do XQuery e uma verificação com XSD.

Com os dados obtidos previamente, o BaseX padroniza-os de modo a devolver um xml com um relatório com estatísticas dos registos clínicos usando o XQuery, como demonstrado na seguinte figura.

```
1. let $xml := element RegistosClinicos {  
2.     for $registro in $json/data/_/Registos/_  
3.     return element RegistoClinico {  
4.         element ID_Registo_Clinico { $registro/ID__Registo__Clinico/text() },  
5.         element ID_Paciente { $registro/ID__Paciente/text() },  
6.         element ID_Profissional { $registro/ID__Profissional/text() },  
7.         element Data_Atendimento { substring-before($registro/Data__Atendimento,  
"T") },  
8.         element Diagnosticos {},  
11.        element Tratamentos {}  
13.    },  
14.    element Pacientes {  
15. },  
16.    element Estatisticas {  
17.        element FaixaEtaria {},  
19.        element PorGenero {},  
21.        element TotalTratamentos {  
$json/data/_/estatisticas/totalTratamentos/text() }  
22.    }  
23. }
```

Figura 7 - XQuery usado para os registos clínicos



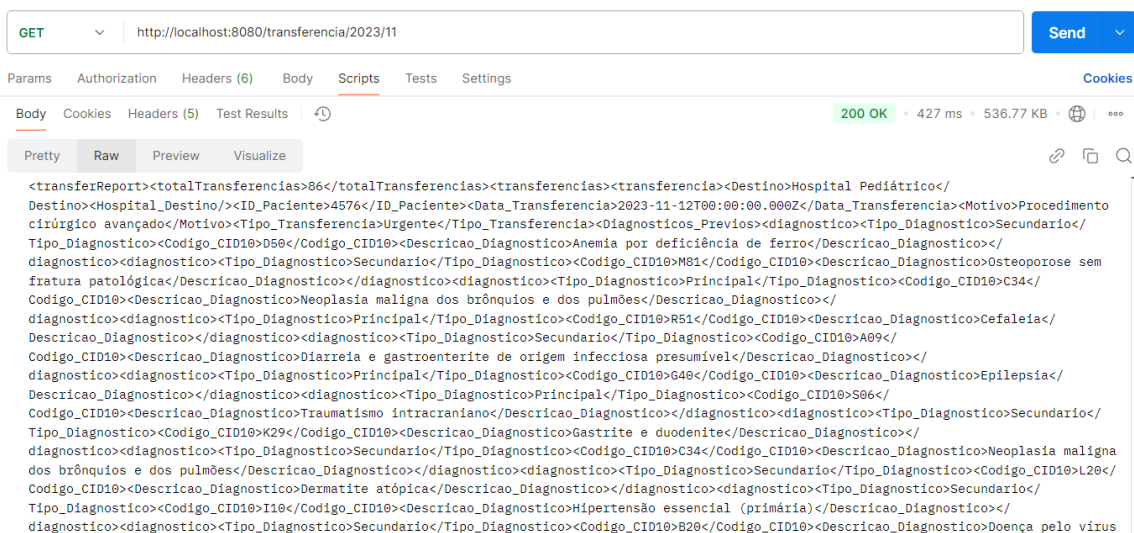
GET http://localhost:8080/getRegistosClinico/2023/11

200 OK • 167 ms • 40.02 KB

```
<RegistosClinicos><RegistoClinico><ID_Registo_Clinico><ID_Registo__Clinico type="number">358</ID_Registo__Clinico></ID_Registo_Clinico><ID_Paciente type="number">7487</ID_Paciente></ID_Paciente><ID_Profissional type="number">192</ID_Profissional></ID_Profissional><Data_Atendimento><Data_Atendimento>2023-11-23T00:00:00.000Z</Data_Atendimento></Data_Atendimento><Diagnosticos><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico></Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>K29</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Gastrite e duodenite</Descricao_Diagnostico></Descricao_Diagnostico></Diagnostico><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico></Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>F32</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Episódios depressivos</Descricao_Diagnostico></Descricao_Diagnostico></Diagnostico><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico></Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>K29</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Gastrite e duodenite</Descricao_Diagnostico></Descricao_Diagnostico></Diagnostico><Tratamentos></RegistoClinico><RegistoClinico><ID_Registo_Clinico><ID_Registo__Clinico type="number">347</ID_Registo__Clinico></ID_Registo_Clinico><ID_Paciente type="number">178</ID_Paciente></ID_Paciente><ID_Profissional type="number">178</ID_Profissional></ID_Profissional><Data_Atendimento><Data_Atendimento>2023-11-20T00:00:00.000Z</Data_Atendimento></Data_Atendimento><Diagnosticos><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico></Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>D50</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Anemia por deficiência de ferro</Descricao_Diagnostico></Descricao_Diagnostico></Diagnostico><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico></Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>T78</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Reações adversas a alimentos não especificados</Descricao_Diagnostico></Descricao_Diagnostico></Diagnostico><Diagnostico><Tipo_Diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico></Tipo_Diagnostico></
```

Figura 8 – Resposta do BaseX com os Registos Clínicos padronizado

Como para os registos clínicos, o BaseX padroniza os dados das transferências e devolve-os num ficheiro xml.



GET http://localhost:8080/transferencia/2023/11

200 OK • 427 ms • 536.77 KB

```
<transferReport><totalTransferencias>86</totalTransferencias><transferencias><transferencia><Destino>Hospital Pediátrico</Destino><Hospital_Destino></ID_Paciente>4576</ID_Paciente><Data_Transferencia>2023-11-12T00:00:00.000Z</Data_Transferencia><Motivo>Procedimento cirúrgico avançado</Motivo><Tipo_Transferencia>Urgente</Tipo_Transferencia><Diagnosticos_Previos><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>D50</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Anemia por deficiência de ferro</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>M81</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Osteoporose sem fratura patológica</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>C34</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Neoplasia maligna dos brônquios e dos pulmões</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>R51</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Cefaleia</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>A09</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Diarreia e gastroenterite de origem infecciosa presumível</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>G40</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Epilepsia</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Principal</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>S06</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Traumatismo intracraniano</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>K29</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Gastrite e duodenite</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>C34</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Neoplasia maligna dos brônquios e dos pulmões</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>L20</Codigo_CID10></Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Dermatite atópica</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>I10</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Hipertensão essencial (primária)</Descricao_Diagnostico></diagnostico><diagnostico><Tipo_Diagnostico>Secundario</Tipo_Diagnostico><Codigo_CID10><Codigo_CID10>B20</Codigo_CID10><Descricao_Diagnostico><Descricao_Diagnostico>Doença pelo vírus
```

Figura 9 – Resposta do BaseX com o relatório de transferências padronizado

XSD

O XSD é um formato de ficheiro que serve para a verificação de ficheiros XML.

Os ficheiros XSD usados verificam a integridade da estrutura e a informação introduzida dos ficheiros XML devolvidos pelo BaseX.

As seguintes figuras demonstram como é estruturada a informação dos relatórios de transferências e registos clínicos, respetivamente:

```
<xs:element name="transferReport">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="totalTransferencias" type="xs:integer"/>
      <xs:element name="transferencias">
        <xs:element name="porMotivo">
          <xs:element name="porTipo">
            </xs:sequence>
          </xs:element>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Figura 10 – extrato simples do .xsd das transferências

```
<xs:element name="RegistosClinicos">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="RegistoClinico" maxOccurs="unbounded">
        <xs:element name="Pacientes">
          <xs:element name="Estatisticas">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="FaixaEtaria">
                  <xs:element name="PorGenero">
                    <xs:element name="TotalTratamentos" type="xs:integer"/>
                  </xs:sequence>
                </xs:element>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figura 11 – Extrato da estrutura do ficheiro .xsd dos registos clínicos

POSTMAN

O Postman foi usado para a verificar o bom funcionamento do projeto e nos diferentes pedidos HTTP. Para tal verificação, basta indicar o URL que se quer realizar o pedido HTTP, o método do pedido e nos parâmetros o mês e o ano pretendido. A figura seguinte mostra a interface do Postman.

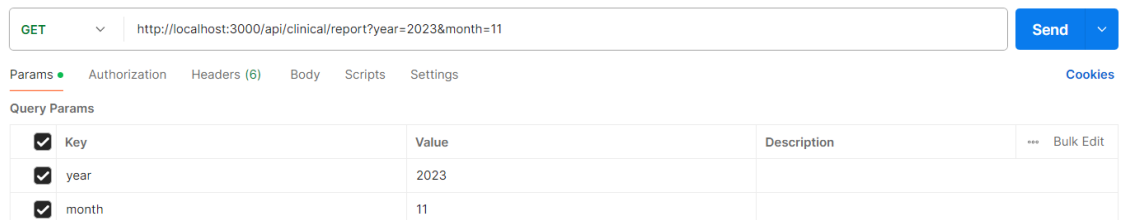


Figura 1212 - Interface do Postman

Encontra-se na pasta entregue no moodle a coleção do Postman que foi exportada pela equipa.

Conclusão

Em suma, o projeto foi finalizado com sucesso, cumprindo os requisitos estipulados desde o início. Todas as funcionalidades foram implementadas conforme o planejado, com um sistema robusto que atende tanto ao lado da BD, consultas e processo de gerar o produto final pretendido, os relatórios em XML.

O grupo manteve um desempenho positivo e mostrou-se interessado durante a execução do projeto, o que foi essencial para a conclusão atempada do mesmo.