

ESCOLA

SUPERIOR

DE

TECNOLOGIA E GESTÃO

POLITÉCNICO DO PORTO

LICENCIATURA EM ENGENHARIA INFORMÁTICA (LEI)

Trabalho de Processamento Estruturado de Informação

Trabalho realizado por:

Artur Pinto nº8230138

Willkie Filho nº8230127

Francisco Oliveira nº8230148

Ano Letivo: 2024/2025

Índice

[Índice de figuras 4](#_Toc186637723)

[Índice de tabelas 8](#_Toc186637724)

[Lista de abreviaturas, siglas e símbolos 9](#_Toc186637725)

[Introdução 10](#_Toc186637726)

[Estrutura XML 11](#_Toc186637727)

[RelatorioRegistosClinicos.xml 12](#_Toc186637728)

[main.xsd 16](#_Toc186637729)

[InfoHospitalRegisto.xsd 18](#_Toc186637730)

[PacientesAtendidos.xsd 20](#_Toc186637731)

[RegistoClinico.xsd 23](#_Toc186637732)

[ResumoMensal.xsd 25](#_Toc186637733)

[ValoresGeraisRegistos.xsd 29](#_Toc186637734)

[RelatorioTransferencias.xml 30](#_Toc186637735)

[main.xsd 34](#_Toc186637736)

[InfoHospitalDestino.xsd 36](#_Toc186637737)

[Transferencias.xsd 37](#_Toc186637738)

[ResumoMensal.xsd 40](#_Toc186637739)

[XSD Gerais 44](#_Toc186637740)

[InfoGeralHospital.xsd 45](#_Toc186637741)

[Diagnosticos.xsd 47](#_Toc186637742)

[Tratamentos.xsd 49](#_Toc186637743)

[Mongo DB 51](#_Toc186637744)

[Modelação 52](#_Toc186637745)

[Registos Clínicos 53](#_Toc186637746)

[Pacientes 57](#_Toc186637747)

[Indexes 59](#_Toc186637748)

[Functions 61](#_Toc186637749)

[updateContactosPacientes 63](#_Toc186637750)

[updateTipoPaciente 65](#_Toc186637751)

[getPacientes 68](#_Toc186637752)

[getRegistosClinicos 69](#_Toc186637753)

[getRelatorioRegistosClinicos 70](#_Toc186637754)

[getResumoMensalRegistosClinicos 71](#_Toc186637755)

[getResumoMensalRegistosClinicosPorGenero 74](#_Toc186637756)

[getHospitaisTransferencia 78](#_Toc186637757)

[getTransferencias 80](#_Toc186637758)

[getRelatorioTransferencias 83](#_Toc186637759)

[getResumoMensalTransferencia 84](#_Toc186637760)

[Postman 86](#_Toc186637761)

[BaseX 87](#_Toc186637762)

[RelatorioRegistosClinicos.xq 88](#_Toc186637763)

[RelatorioTransferencias.xq 91](#_Toc186637764)

[Conclusão 94](#_Toc186637765)

[Referências 95](#_Toc186637766)

# Índice de figuras

[Figura 1 - namespaces RelatorioRegistosClinicos.xml 12](#_Toc186637583)

[Figura 2 - elemento hospital 12](#_Toc186637584)

[Figura 3 elemento lista\_pacientes 13](#_Toc186637585)

[Figura 4 - elemento lista\_registos 14](#_Toc186637586)

[Figura 5 - elemento resumo\_mensal 15](#_Toc186637587)

[Figura 6 - namespace main.xsd 16](#_Toc186637588)

[Figura 7 - element relatorio\_clinico 16](#_Toc186637589)

[Figura 8 - elemento lista\_pacientes 17](#_Toc186637590)

[Figura 9 - elemento lista\_registos 17](#_Toc186637591)

[Figura 10 - schema InfoHospitalRegisto.xsd 18](#_Toc186637592)

[Figura 11 - complexType hospitalType 18](#_Toc186637593)

[Figura 12 - simpleType mesType 19](#_Toc186637594)

[Figura 13 - simpleType ano\_fiscType 19](#_Toc186637595)

[Figura 14 - schema PacientesAtendidos.xsd 20](#_Toc186637596)

[Figura 15 - complexType pacienteType 20](#_Toc186637597)

[Figura 16 - simpleTypes PacientesAtendidos.xsd 21](#_Toc186637598)

[Figura 17 - elemento contacto 22](#_Toc186637599)

[Figura 18 - simpleType emailType 22](#_Toc186637600)

[Figura 19 - schema RegistoClinico.xsd 23](#_Toc186637601)

[Figura 20 - complexType registoClinicoType 24](#_Toc186637602)

[Figura 21 - simpleType idType 24](#_Toc186637603)

[Figura 22 - schema ResumoMensal.xsd 25](#_Toc186637604)

[Figura 23 - complexType resumoMensalType 25](#_Toc186637605)

[Figura 24 - simpleType totType 25](#_Toc186637606)

[Figura 25 - Elementos gerais do ResumoMensal.xsd 26](#_Toc186637607)

[Figura 26 - elemento genero 26](#_Toc186637608)

[Figura 27 - elemento faixas\_etarias 27](#_Toc186637609)

[Figura 28 - elemento faixa\_etaria 27](#_Toc186637610)

[Figura 29 - elemento especialidades 28](#_Toc186637611)

[Figura 30 - simpleType idadeType 28](#_Toc186637612)

[Figura 31 - elemento especialidade 28](#_Toc186637613)

[Figura 32 - schema ValoresGeraisRegistos.xsd 29](#_Toc186637614)

[Figura 33 - simpleType generoType 29](#_Toc186637615)

[Figura 34 - simpleType especialidadeType 29](#_Toc186637616)

[Figura 35 – Namespaces RelatorioTransferencias.xml 30](#_Toc186637617)

[Figura 36 – Elemento Hospitais 30](#_Toc186637618)

[Figura 37 – Exemplo do elemento lista\_transferencias 32](#_Toc186637619)

[Figura 38 – Exemplo do elemento tratamentos 32](#_Toc186637620)

[Figura 39 – Exemplo do elemento resumo\_mensal 33](#_Toc186637621)

[Figura 40 – Declaração dos namespaces do main.xsd 34](#_Toc186637622)

[Figura 41 - Declaração do elemento relatorio\_transferencias no esquema main.xsd 34](#_Toc186637623)

[Figura 42 - Declaração do elemento lista\_transferencias no esquema main.xsd. 35](#_Toc186637624)

[Figura 43 - Declaração do elemento hospitais no esquema main.xsd 35](#_Toc186637625)

[Figura 44 - Declaração do namespace e importação do xsd necessário 36](#_Toc186637626)

[Figura 45 - Definifção do complexType hospitalType 36](#_Toc186637627)

[Figura 46 - Declaração do namespace e importação de outros xsd necessários. 37](#_Toc186637628)

[Figura 47 - Regras da construção do elemento transferenciaType 37](#_Toc186637629)

[Figura 48 – Definição do simpleType tipoTransferenciaType 38](#_Toc186637630)

[Figura 49 - Definição do simpleType idType 38](#_Toc186637631)

[Figura 50 - Definição do simpleType motivoType 38](#_Toc186637632)

[Figura 51 - Regras da construção do elemento relatorios\_medicos 39](#_Toc186637633)

[Figura 52 - Regras da construção do elemento relatorio\_medicos 39](#_Toc186637634)

[Figura 53 - Definição do simpleType especialidadeType 39](#_Toc186637635)

[Figura 54 - Declaração do namespace e importação dos xsd necessários 40](#_Toc186637636)

[Figura 55 - Regras da construção do elemento tot-transferencias e do elemento complexType resumoMensalType 40](#_Toc186637637)

[Figura 56 - Regras da construção dos simpleTypes tot\_type, nomeType, idType 41](#_Toc186637638)

[Figura 57 - Regras da construção dos elementos motivos e motivo 42](#_Toc186637639)

[Figura 58 - Regras da construção dos elementos tipos e tipo 42](#_Toc186637640)

[Figura 59 - Regras da construção dos elementos tipos e tipo 43](#_Toc186637641)

[Figura 60 - schema InfoGeralHospital.xsd 45](#_Toc186637642)

[Figura 61 - elementos InforGeralHospital.xsd 45](#_Toc186637643)

[Figura 62 - atributo id InfoGeralHospital.xsd 46](#_Toc186637644)

[Figura 63 - schema Diagnosticos.xsd 47](#_Toc186637645)

[Figura 64 - SimpleTypes Diagnosticos.xsd 47](#_Toc186637646)

[Figura 65 - elementos Diagnosticos.xsd 48](#_Toc186637647)

[Figura 66 - schema Tratamentos.xsd 49](#_Toc186637648)

[Figura 67 - simpleTypes Tratamentos.xsd 49](#_Toc186637649)

[Figura 68 - elementos Tratamentos.xsd 50](#_Toc186637650)

[Figura 69 - Sort dos registos e tipo diagnostico 53](#_Toc186637651)

[Figura 70 - group dos Diagnosticos 53](#_Toc186637652)

[Figura 71 - limit de registos clinicos 54](#_Toc186637653)

[Figura 72 - Lookup dos profissionais 54](#_Toc186637654)

[Figura 73 - unwind Dados dos Profissionais 55](#_Toc186637655)

[Figura 74 - loookup dos tratmanetos 55](#_Toc186637656)

[Figura 75 - Project dos Registos Clinicos 56](#_Toc186637657)

[Figura 76 - out Registos Clinicos 56](#_Toc186637658)

[Figura 77 - lookup dos registos clinicos nos pacientes 57](#_Toc186637659)

[Figura 78 - lookup das transferencias nos pacientes 57](#_Toc186637660)

[Figura 79 - project dos pacientes 58](#_Toc186637661)

[Figura 80 - Indexes Pacientes 59](#_Toc186637662)

[Figura 81 - Indexes Profissionais 59](#_Toc186637663)

[Figura 82 - Index Registos Clinicos 59](#_Toc186637664)

[Figura 83 - Index Tratamentos 60](#_Toc186637665)

[Figura 84 - Index Transferências 60](#_Toc186637666)

[Figura 85 - Verificação pelo ano e mes 62](#_Toc186637667)

[Figura 86 - Erro status 500 62](#_Toc186637668)

[Figura 87 - Conexão coleção updateContactosPacientes 63](#_Toc186637669)

[Figura 88 - Constantes para a data minima e maxima do registo 63](#_Toc186637670)

[Figura 89 - codigo para atualizar telefone dos pacientes 63](#_Toc186637671)

[Figura 90 - código para atualizar email dos pacientes 64](#_Toc186637672)

[Figura 91 - Retorno atualizacao dos contactos 64](#_Toc186637673)

[Figura 92 - Conexão e variáveis do updateTipoPaciente 65](#_Toc186637674)

[Figura 93 - constante pacientes e arrays 65](#_Toc186637675)

[Figura 94 - Preenchimento dos arrays de tipo de paciente 66](#_Toc186637676)

[Figura 95 - Atualização do campo de tipo de paciente 67](#_Toc186637677)

[Figura 96 - Retorno atualização dos pacientes 67](#_Toc186637678)

[Figura 97 - execução funções no getPacientes 68](#_Toc186637679)

[Figura 98 - codigo do getPacientes 68](#_Toc186637680)

[Figura 99 - codigo para pesquisar pelos registos clinicos 69](#_Toc186637681)

[Figura 100 - codigo do getRelatorioRegistosClinicos 70](#_Toc186637682)

[Figura 101 - constantes do genero 71](#_Toc186637683)

[Figura 102 - processo de guardar as faixas etarias 72](#_Toc186637684)

[Figura 103 - Total de atendiemntos e tratamentos de cada género 73](#_Toc186637685)

[Figura 104 - Varaivel Resumo Mensal dos registos clinicos 73](#_Toc186637686)

[Figura 105 - Variaveis do getResumoMensalRegistosClinicosPorGenero 74](#_Toc186637687)

[Figura 106 - match faixa\_etaria 74](#_Toc186637688)

[Figura 107 - unwind Registos Clinicos getResumoMensalRegistosClinicosPorGenero 75](#_Toc186637689)

[Figura 108 - Set Aggreate getResumoMensalRegistosClinicosPorGenero 75](#_Toc186637690)

[Figura 109 - aggreate getResumoMensalRegistosClinicosPorGenro addFields 75](#_Toc186637691)

[Figura 110 - Group especialidades getResumoMensalRegistosClinicosPorGenero 76](#_Toc186637692)

[Figura 111 - addFiedls 2 getResumoMensalRegisotsClinicosPorGenero 76](#_Toc186637693)

[Figura 112 - group 3 getResumoMensalRegistosClinicosPorGenero 77](#_Toc186637694)

[Figura 113 - retorno getResumoMensalRegistosClinicosPorGenero 77](#_Toc186637695)

[Figura 114 - pipeline do getHospitaisTransferencia 78](#_Toc186637696)

[Figura 115 - resultado pipeline getHospitaisTransferencia 79](#_Toc186637697)

[Figura 116 - Definifição da data de inicio e da data de fim, e o inicio da pipeline 81](#_Toc186637698)

[Figura 117 – definição do primeiro $project 81](#_Toc186637699)

[Figura 118 - Fim do primeiro project e inicio do segundo 82](#_Toc186637700)

[Figura 119 - Fim do segundo project e fim da function 82](#_Toc186637701)

[Figura 120 - codigo getRelatorioTransferencias 83](#_Toc186637702)

[Figura 121 - Function getResumoMensalTransferencias 84](#_Toc186637703)

[Figura 122 - Retorno do resultado e retorno de mensagem de erro caso necessário 85](#_Toc186637704)

[Figura 123 - Postman 86](#_Toc186637705)

[Figura 124 - Declaração dos namespace de RelatorioRegistosClinicos.xq 88](#_Toc186637706)

[Figura 125 - Configuração do Endpoint REST de RelatoriosRegistosClinicos.xq 88](#_Toc186637707)

[Figura 126 - Construção do URL de requisição de RelatorioRegistosClinicos.xq 88](#_Toc186637708)

[Figura 127 - Requisição HTTP RelatorioRegistosClinicos.xq 88](#_Toc186637709)

[Figura 128 - Pequeno trecho do elemento principal RelatorioRegistosClinicos.xq 89](#_Toc186637710)

[Figura 129 - Elemento hospital RelatorioRegistosClinicos.xq 89](#_Toc186637711)

[Figura 130 - Elemento lista\_pacientes RelatorioRegistosClinicos.xq 89](#_Toc186637712)

[Figura 131 - Elemento lista\_ registos RelatorioRegistosClinicos.xq 90](#_Toc186637713)

[Figura 132 - Elemento resumo\_mensal RelatoriosRegistosClinicos.xq 90](#_Toc186637714)

[Figura 133 - Declaração dos namespaces 91](#_Toc186637715)

[Figura 134 - Declaração do Endpoint REST 91](#_Toc186637716)

[Figura 135 - Construção do URL 91](#_Toc186637717)

[Figura 136 - Requisição HTTP 92](#_Toc186637718)

[Figura 137 - Pequeno excerto do elemento raiz 92](#_Toc186637719)

[Figura 138 - Elemento hospitais 92](#_Toc186637720)

[Figura 139 - Elemento lista\_transferencias 93](#_Toc186637721)

[Figura 140 - Elemento resumo\_mensal 93](#_Toc186637722)

# Índice de tabelas

[Tabela 1 - Tabela de Abreviatura, siglas e símbolos 6](#_Toc186543320)

# Lista de abreviaturas, siglas e símbolos

|  |  |
| --- | --- |
| Sigla | Descrição |
| XML | eXtensible Markup Language |
| XSD | XML Schema |
| ID | Identifier |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| URL | Uniform Resource Locator |
| HTTP | Hypertext Transfer Protocol |

Tabela - Tabela de Abreviatura, siglas e símbolos

# Introdução

O *MedSync* é um consórcio de hospitais que pretende utilizar um processo para padronizar e partilhar os dados clínicos entre os hospitais parceiros. Este processo é composto por três etapas principais:

* Extração de Dados: Os dados originais são extraídos das bases de dados dos hospitais e armazenados numa base de dados *MongoDB*;
* Disponibilização de Dados via *API REST*: Após a integração no *MongoDB*, os dados são disponibilizados em formato *JSON* através de uma *API REST*;
* Conversão de Dados para *XML* com *XQuery* (a partir da ferramenta *BaseX*): 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*.

# Estrutura XML

Este tópico iremos abordar da estrutura XML que utilizamos, onde a mesma é constituída por dois ficheiros xml, sendo eles:

* RelatorioRegistosClinicos.xml, que é um relatório sobre os registos clínicos realizados num determinado mês;
* RelatorioTransferencias.xml, que é um relatório sobre as transferências, realizadas num determinado mês.

Para organizar melhor o relatório, dividimos este tópico em três subtópicos onde os dois primeiros falam sobre a estrutura dos xmls, e o utlimo fala dos xsd gerais, ou seja, os xsd que são utilizados em ambos os xml. Também metemos subtópicos em cada um dos subtópicos da “Estrutura XML”, onde falamos de cada um dos xsds que estão associados a cada um dos xml.

## RelatorioRegistosClinicos.xml

Este xml serve para representar o relatório dos registos clínicos para um mês e ano específico, onde a estrutura do mesmo está dividida nestes quatro elementos, sendo eles:

* “hospital”, que guarda a informação do hospital, juntamente com o mês e ano fiscal;
* “lista\_pacientes”, guarda todos os pacientes cujo mês da sua data de registo é a mesma do relatorio;
* “lista\_registos”, guarda todos os registos clínicos efetuados no mês do relatório;
* Uma imagem com texto, captura de ecrã, Tipo de letra

  Descrição gerada automaticamente“resumo\_mensal”, resumo dos registos clínicos, onde é indicado o total de atendimentos realizados no mês e para cada género (masculino e feminino) é indicado o total de tratamentos realizados e caso crónicos detetados, por faixa etária e especialidade médica.

Figura - namespaces RelatorioRegistosClinicos.xml

Como dá para observar na imagem acima, começamos por definir as namespace necessárias, para a criação do xml, como por exemplo a namespace “https://www.medsync.com/listaPacientes” para os dados dos pacientes (xmlns:p) . Para além disso associamos o documento a uma schema XML externa, a “main.xsd”, com o uso do “xsi:schemaLocation”.

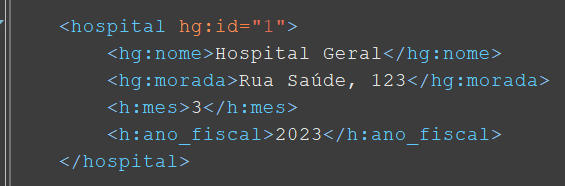


Figura - elemento hospital

Acima dá para observar o elemento “hospital”, que é o 1º elemento deste xml e armazena a informação do hospital (id que é um atributo, nome e morada), para além de conter o mês e o ano fiscal (“h:ano\_fiscal”) do relatório.

Na imagem abaixo dá para observar o elemento “lista\_pacientes”, esse elemento é capaz de armazenar todos os pacientes do relatório, onde os dados do paciente são armazenados no elemento “paciente”. O elemento “lista\_pacientes” é capaz de armazenar 0 ou mais elementos “paciente”, dentro do mesmo. O elemento paciente é composto pelo atributo “id” e pelos seguintes elementos:

* nome;
* data\_nascimento;
* genero;
* contacto, que contêm os elementos telefone e email;
* tipo\_paciente.

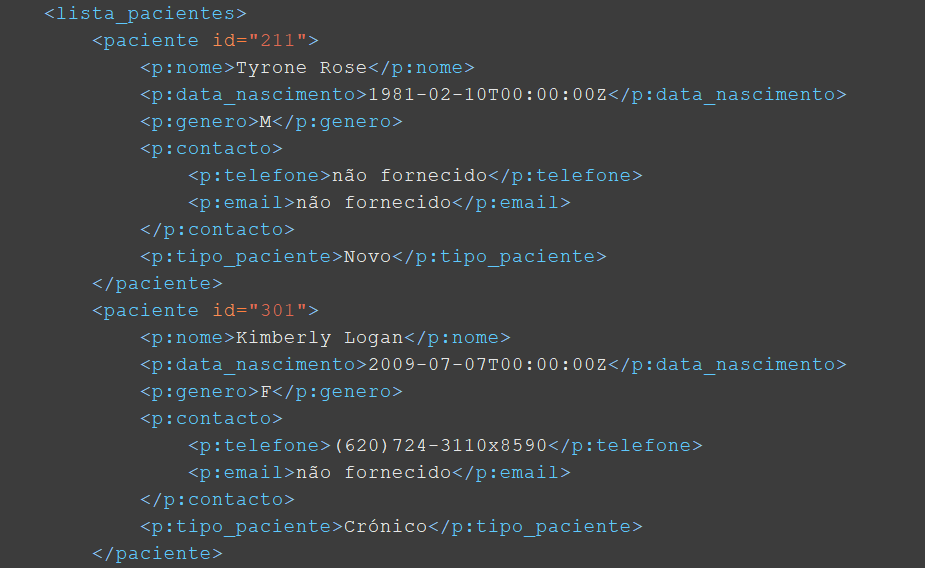


Figura elemento lista\_pacientes

A imagem abaixo mostra o 3º elemento do XML, o elemento “lista\_registos”, que têm o mesmo funcionamento que o elemento “lista\_pacientes” o que muda entre os dois é que enquanto a “lista\_pacientes” mostra os dados dos pacientes este elemento mostra os dados dos registos clínicos. Os registos clínicos, são representados pelo elemento “registo\_clinico”, que contém o atributo “codigo\_atend”, que armazena o código do registo clínico e os seguintes elementos:

* data;
* id\_medico;
* diagnosticos, este elemento opcional, ou seja, só aparece quando o registo clínico possui diagnósticos. Este elemento quando existe armazena diagnósticos do registo clínico, armazenado cada diagnóstico no elemento “diagnostico”, sendo que esse elemento contém o atributo “Codigo\_CID10” e os seguintes elementos:
  + tipo\_diagnostico;
  + descricao.
* tratamentos, tanto como o elemento diagnóstico este elemento também é opcional e só aparece quando o registo clínico possui tratamentos, cada tratamento é representado pelo elemento tratamento, que possui o atributo “cod\_tratamento” e o elemento tipo.

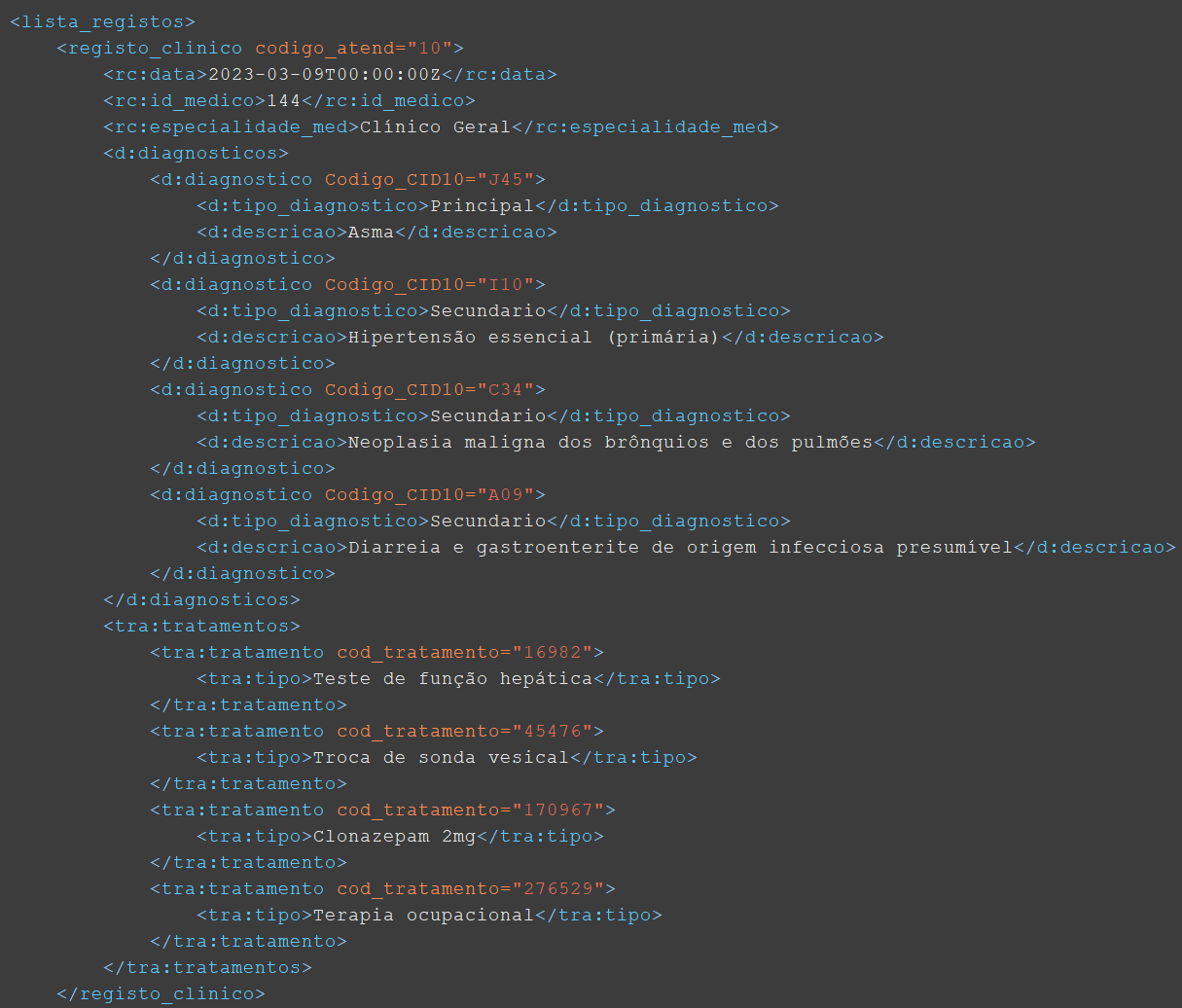


Figura - elemento lista\_registos

Na imagem abaixo é possível observar o 4º e último elemento deste xml, o elemento “resumo\_mensal” este elemento contém 2 elementos, sendo eles:

* tot\_atendimentos;
* genero, este elemento aparece obrigatoriamente 2 vezes, pois cada um contém a informações referente a cada genero (masculino e feminino), cada um contém o atributo id\_genero, e possui os seguinte elementos:
  + tot\_tratamentos;
  + faixas\_etaria, este elemento guarda todas as faixa\_etarias do elemento genero, nos elementos “faixa\_etaria” onde esse elemento armazena os dados de cada faixa etaria, onde essa informação é representada, nos seguintes elementos:
    - idades;
    - tot\_tratamentos;
    - tot\_cronicos;
    - especialidades, este elemento armazena todas as especialidades médicas que fizeram atendimentos aos pacientes da faixa etária representado cada uma das especialidades no elemento “especialidade”, que contém os seguintes campos:
      * nome\_especialidade;
      * tot\_tratamentos;
      * tot\_cronicos.

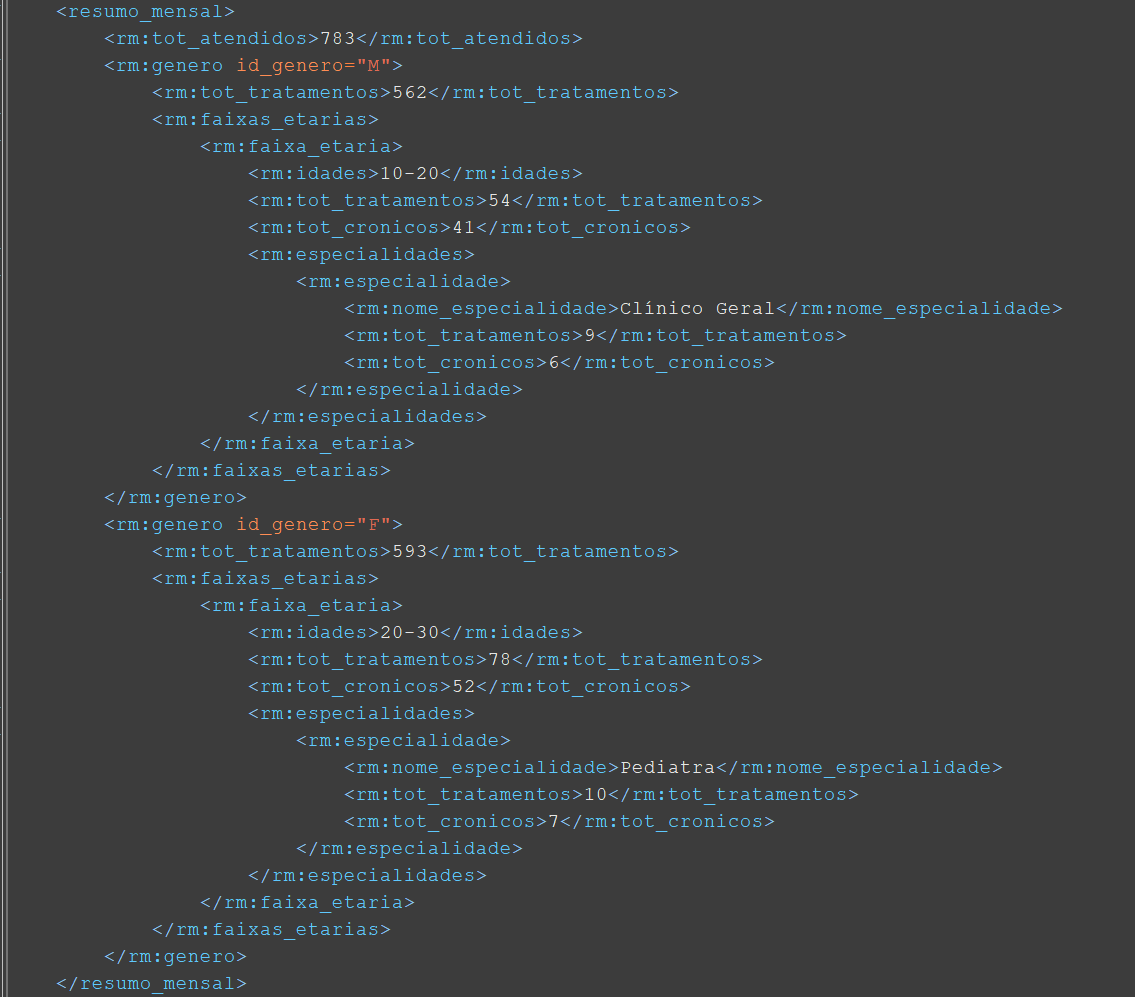


Figura - elemento resumo\_mensal

### main.xsd

Este xsd foi criado com a intenção, de juntar todos os xsd que fazem parte do RelatorioRegistosClinicos.xml, num único xsd e definir a estrutura do mesmo xml.

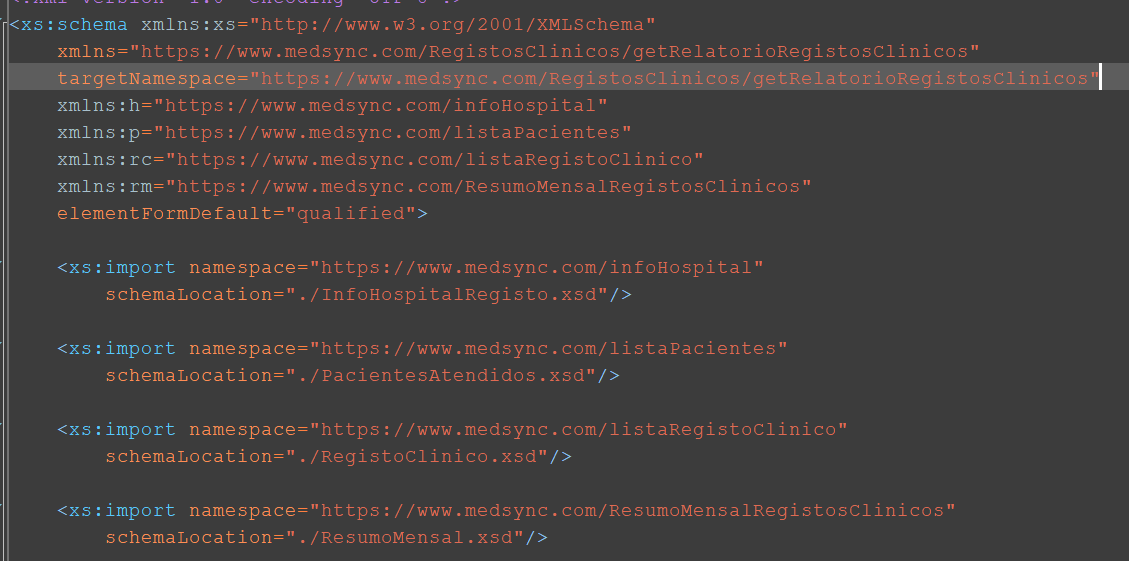
 Como dá para observar na imagem abaixo, no main.xsd começamos por declarar os namespace e importar os mesmos.

Figura - namespace main.xsd

Depois criamos o elemento relatorio\_clinico, como dá para observar na imagem abaixo, que é o elemento principal deste xsd. O mesmo possui um complexType e dentro do mesmo é definida por 4 elementos, sendo eles:

* “hospital”, que é do tipo “h:hospitalType”;
* “lista\_pacientes”, que é uma referencia de um elemento com o mesmo nome;
* “lista\_registos”, que é uma referencia de um elemento com o mesmo nome;
* Uma imagem com texto, captura de ecrã, Tipo de letra

  Descrição gerada automaticamente“resumo\_mensal”, que é do tipo “rm:resumoMensalType”.

Figura - element relatorio\_clinico

Os 4 elementos que estão apresentados no “RelatorioRegistosClinicos.xml”.

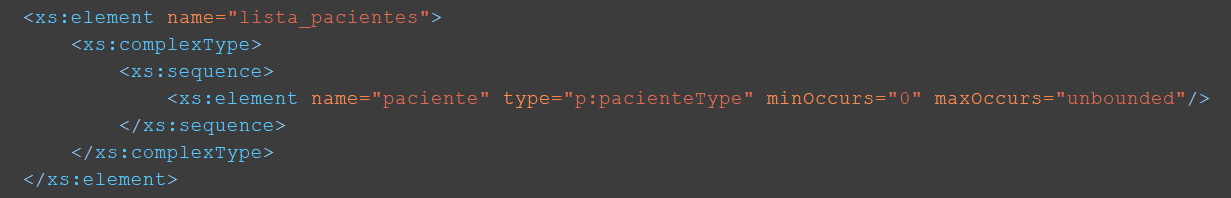
O elemento que dá para observar na imagem abaixo, é o “lista\_pacientes”, o mesmo elemento referencia na imagem acima, só que aqui nos definimos a estrutura dele, onde definimo-lo como um complexType, com uma sequencia de um elemento, o elemento “paciente”, que é do tipo “p:pacienteType” e pode ocorrer no minimo 0 vezes, ou infinitas vezes.

Figura - elemento lista\_pacientes

Por fim temos o elemento “lista\_registos”, que tem um tipo complexo, com uma sequencia de 1 elemento, com o elemento “registo\_clinico”, que é do tipo “rc:registoClinicoType” e esse elemento pode ocorrer no minimo 0 até no máximo infinitas vezes.

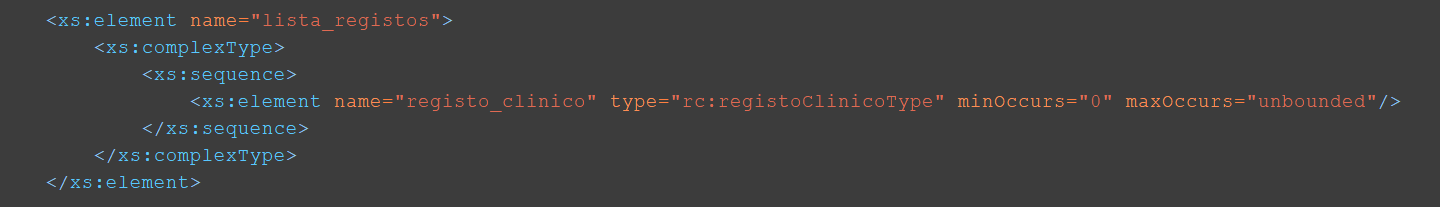


Figura - elemento lista\_registos

### InfoHospitalRegisto.xsd

Neste xsd é feita a estruturação do elemento “hospital” do “RelatorioRegistosClinicos.xml”.

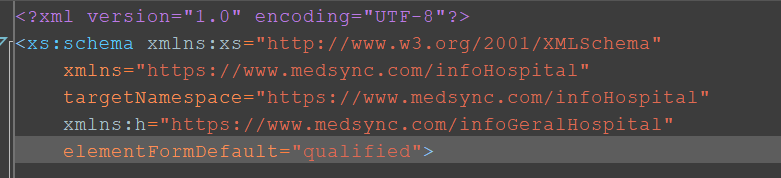
 Como dá para ver na imagem abaixo, começamos por definir o targetNamespace e o xmlns deste xsd e inicializar o namespace “xmlns:h="<https://www.medsync.com/infoGeralHospital>"”.

Figura - schema InfoHospitalRegisto.xsd

Depois definimos o complexType “hospitalType”, que possui uma sequencia com 4 elementos, sendo dois deles referência para o elementos do xsd “InfoGeralHospital.xsd”, sendo eles os elementos “h:nome” e “h:morada”, depois temos o elemento “mes”, que é do tipo “mesType” e o elemento “ano\_fiscal”, que é do tipo “ano\_fiscType”. O complexType ainda possui um atributo, sendo ele o “h:id”, que também pertence ao xsd “InfoGeralHospital.xsd” e que é de uso obrigatório.

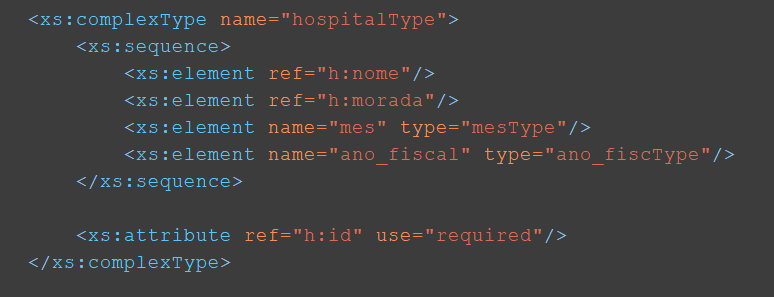


Figura - complexType hospitalType

Na imagem abaixo é possível observar o simpleType “mesType”, que possui uma restriction para definir que vai ser do tipo “xs:int”, ou seja, vai ser um número inteiro e só poderá ser um valor entre 1 e 12, incluindo os mesmos.

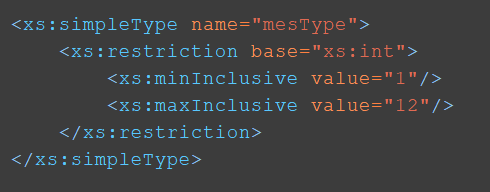


Figura - simpleType mesType

O “ano\_fiscType” é um sumpleType de tipo inteiro, que aceita apenas valores inteiros entre 1900 e 2025.

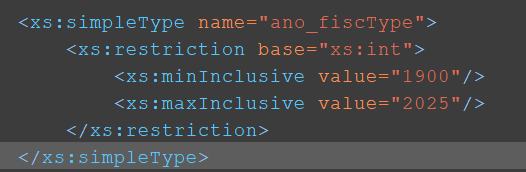


Figura - simpleType ano\_fiscType

### PacientesAtendidos.xsd

Neste xsd é definido o complexType pacienteType, que vai ser a estrutura principal deste xsd pois ira definir como é constituído um paciente no RelatorioRegistosClinicos.xml.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Como dá para observar na imagem abaixo, começamos por definir um targetNamespace ao “PacientesAtendidos.xsd” e depois definir e importar o xsd “ValoresGeraisRegistos.xsd”.

Figura - schema PacientesAtendidos.xsd

Após definir o targetNamespace, criamos o complexType “pacienteType”, que vai definir a estrutura de qualquer elemento do seu tipo. Sendo que o mesmo possui uma sequência com 5 elementos, começando com o elemento “nome”, que é do tipo “nomeType”, de seguida temos o elemento “data\_nascimento”, que vai ser do tipo “xs:dateTime”, ou seja, esse elemento vai receber uma data e hora, depois temos o elemento “genero”, que é do tipo “generoType”, temos também o elemento “contacto” que está referenciado, ou seja, o elemento “contacto” já foi criado em algum lado e ao referenciar estamos a dizer que aquele elemento vai ser igual ao elemento “contacto” já criado, e ainda temos o elemento “tipo\_paciente” que é do tipo “tipo\_pacienteType”. Falta ainda ressaltar que o “pacienteType” ainda possui um atributo id que é do tipo “id\_type” e que é de uso obrigatório.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - complexType pacienteType

Na imagem abaixo é possível observar 4 simpleType, que foram utilizados no complexType “pacienteType”, sendo eles:

* “idType”, este simpleType é do tipo “xs:int” e ele só permite receber valores maiores ou iguais a 0;
* “nomeType”, este simpleType é do tipo “xs:string” e ele só pode receber no máximo 100 caracteres e um limite mínimo de 1 caracter;
* Uma imagem com texto, captura de ecrã, Tipo de letra, menu

  Descrição gerada automaticamente“tipo\_pacienteType”, que também é do tipo “xs:string” e possui 3 enumerations, o que faz com ele só possa aceitar um destes 3 valores “Novo”, “Regular” ou “Crónico”.

Figura - simpleTypes PacientesAtendidos.xsd

Na imagem abaixo podemos observar o elemento “contacto” o mesmo que é referenciado no “pacienteType”, e podemos observar que ele possui um complexType, com uma sequência de dois elementos, sendo esses elementos os seguintes:

* “telefone”, este elemento é do tipo xs:string”;
* “email”, que é do tipo “emailType”.

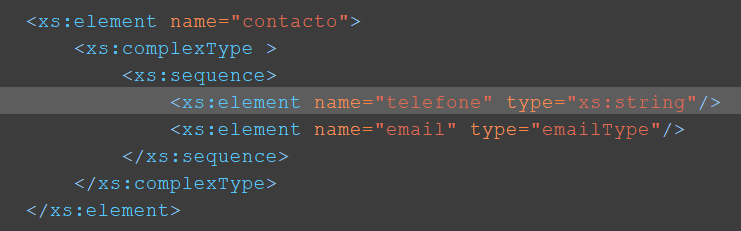


Figura - elemento contacto

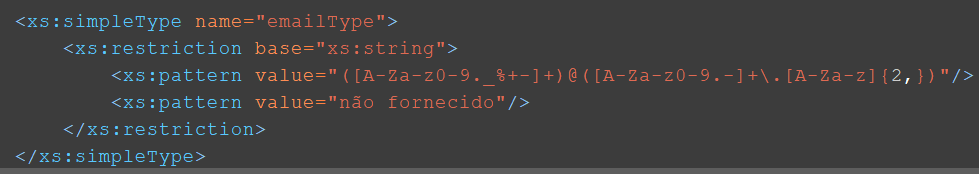
Por fim, para terminar este XSD, temos na imagem abaixo o simpleType emailType, que é restringido ao tipo xs:string e possui dois pattern, cada um indicando quais são os valores possíveis para o e-mail ser considerado válido. O primeiro pattern (<xs:pattern value="([A-Za-z0-9.\_%+-]+)@([A-Za-z0-9.-]+\.[A-Za-z]{2,})"/>) define que um e-mail é válido quando, antes do “@” (([A-Za-z0-9.\_%+-]+)), há uma ou mais ocorrências de letras, números, pontos (.), sublinhados (\_), porcentagens (%), sinais de mais (+) ou hífens (-). Em seguida, o e-mail deve obrigatoriamente conter o caractere “@”. Após o “@” (([A-Za-z0-9.-]+\.[A-Za-z]{2,})), devem existir um ou mais caracteres que sejam letras, números, pontos (.) ou hífens (-), seguidos de um ponto (.). Após o ponto, devem existir ao menos dois caracteres alfabéticos ([A-Za-z]{2,}), que correspondem ao domínio de nível superior (TLD), como .com, .org, etc. Já o segundo pattern (<xs:pattern value="não fornecido"/>) permite que, caso o valor não seja um e-mail válido, ele possa ter exatamente o valor “não fornecido”.

Figura - simpleType emailType

### RegistoClinico.xsd

Este xsd serve para definir a estrutura xml de um registo clínico.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Como dá para observar na imagem abaixo começamos por definir o namespace do xsd e por definir e importar os xsd “ValoresGeraisRegistos.xsd”, “Tratamentos.xsd” e “Diangnosticos.xsd”.

Figura - schema RegistoClinico.xsd

Depois criamos o complexType “registoClinicoType”, que possui uma sequência de 5 elementos, sendo eles:

* “data”, que é do tipo dateTime;
* “id\_medico”, que é do tipo “idType”;
* “especialidade\_med”, que é do tipo “vg:especialidadeType”;
* “d:diagnostico”, que é uma referencia para o elemento diagnostico do xsd, “Diagnosticos.xsd” e o elemento ocorre 0 ou mais vezes;
* “t:tratamentos”, é uma referencia para o elemento “tratamentos” do xsd “Tratamentos.xsd” e ocorre 0 ou mais vezes.

O complexType ainda contêm o atributo “código\_atend”, que é do tipo “idType” e é de uso obrigatório.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Na imagem abaixo é possível observar o simpleType “idType”, que é restrito para ser do tipo inteiro e só aceitar valores maiores ou iguais a zero.

Figura - complexType registoClinicoType

Uma imagem com texto, Tipo de letra, captura de ecrã, número

Descrição gerada automaticamente

Figura - simpleType idType

### ResumoMensal.xsd

Este xsd é responsável por definir a estrutura do resumo mensal do “RelatorioRegistosClinicos.xml”.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteMais uma vez como em todos os xsd começamos por especificar o targetNamespace do mesmo.

Figura - schema ResumoMensal.xsd

Após isso criamos o complexType “resumoMensalType”, que vai ser o tipo “principal” deste xsd. Este complexType possui uma sequência com 2 elementos sendo eles:

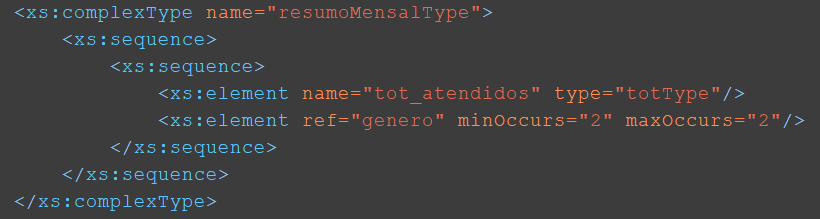
* “tot\_atendidos”, que é do tipo “tot\_type” e indica o total de pacientes atendidos;
* “genero” este elemento é uma referencia do elemento “genero” e ele vai ocorrer obrigatoriamente duas vezes, dentro do resumo Mensal.

Figura - complexType resumoMensalType

Na imagem abaixo dá para observar o simpleType “totType” ele é restringido para ser do tipo “xs:int” e só aceita valores iguais ou superiores a 0.

Uma imagem com texto, Tipo de letra, captura de ecrã, design

Descrição gerada automaticamente

Figura - simpleType totType

Depois criamos dois elementos os elementos “tot\_tratamentos” e “tot\_cronicos”, onde ambos são do tipo “tot\_type” nós definimos ele sem nenhum complexType ou simpleType, pois como vamos eles vão aparecer em mais do que uma sequência decidimos criar já os elementos para depois serem apenas referenciados, pois desta forma caso seja preciso alterar alguma coisa em alguns destes elementos só termos de alterar num único sítio.

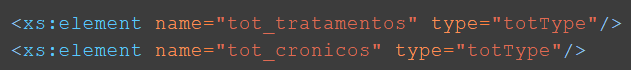


Figura - Elementos gerais do ResumoMensal.xsd

Na imagem abaixo está o elemento “género” que é definido como um complexType, que possui uma sequência com 2 atributos, sendo eles:

* “tot\_tratamentos, que é uma referência para o elemento criado anteriormente;
* “faixas\_etarias”, que é uma referência para o elemento com o mesmo nome.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteO elemento “género” ainda possui um atributo, chamado “id\_genero”, que é do tipo “generoType” e é de preenchimento obrigatório.

Figura - elemento genero

Na imagem abaixo é possível observar o elemento “faixas\_etarias”, que foi referenciado na imagem acima, onde o mesmo é definido como um complexType com uma sequência de um apenas um elemento, o elemento “faixa\_etaria”, que é referencia para outro elemento e ainda é definido que o mesmo pode ocorrer zero ou mais vezes.

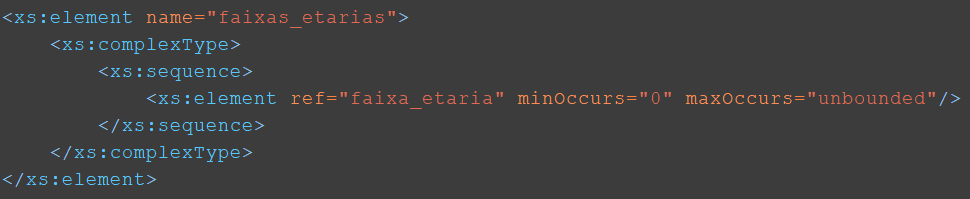
 Na imagem abaixo é possível observar o elemento “faixa\_etaria” que é definido como um complexType com uma sequência de 4 elementos, sendo eles:

Figura - elemento faixas\_etarias

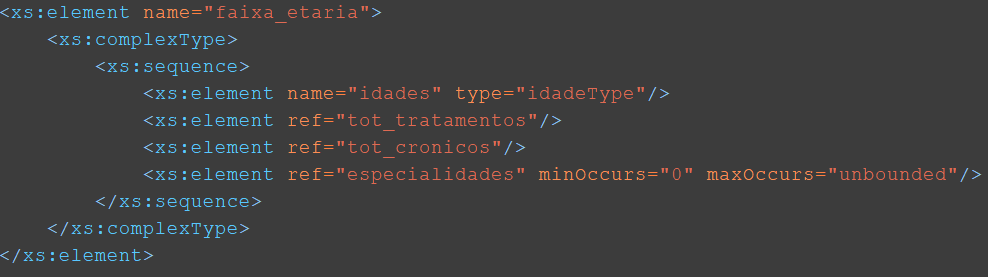
* “idades”, que é do tipo “idadeType” e define as idades da faixa etaria em específico;
* “tot\_tratamentos”, que é uma referência para o elemento “tot\_tratamentos”;
* “tot\_cronicos”, que é uma referência para o elemento “tot\_cronicos”;
* “especialidades”, que é uma referência para o elemento “especialidades”, e o mesmo pode ocorrer 0 ou mais vezes.

Figura - elemento faixa\_etaria

Abaixo é possível observar o simpleType “idadeType”, que foi restringido para ser uma string (“xs:string”), e que possui um pattern, onde ele vai aceitar na 1º parte do mesmo, “([0-9]|[1-9][0-9]|100)”, receber um numero de 0 até 9, “[0-9]“ ou dois números onde o 1º vai de 1 até 9 e o segundo de 0 até 9, “[1-9][0-9]“, ou então o número 100, depois na segunda parte do parttern é obrigatório meter um hífen (-) e depois a 3º parte segue exatamente a mesma logica da 1º.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Também criamos o elemento “especialidades”, que é definido como um complexType, e que possui uma sequência de 1 elemento, sendo ele o uma referência para o elemento “especialidade” e que pode ocorrer 0 ou mais vezes.

Figura - elemento especialidades

Figura - simpleType idadeType

Na imagem abaixo é possível observar o elemento especialidade referenciado abaixo, que é definido como um complexType, com uma sequência de 3 elementos, sendo eles:

* “nome\_especialidade”, que é do tipo “vg:especialidadeType”, que e que indica o nome da especialidade;
* “tot\_tratamentos”, que é uma referência para o elemento “tot\_tratamentos”;
* Uma imagem com texto, captura de ecrã, Tipo de letra

  Descrição gerada automaticamente“tot\_cronicos”, que é uma referência para o elemento “tot\_cronicos”.

Figura - elemento especialidade

### ValoresGeraisRegistos.xsd

Este xsd, serve para guardar simpleType, que podem ser reutilizados mais do que um xsd, do RelatorioRegistosClinicos.xml.

Como em todos os xsd, também começamos por especificar o namespace deste xsd.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - schema ValoresGeraisRegistos.xsd

Após isso definimos o simpleType deste xsd, o “generoType”, que é restringido para ser do tipo string e possui duas enumerations, onde a mesma só permite o “generoType” assumir dois valores “M” ou “F”.

Uma imagem com texto, Tipo de letra, captura de ecrã, design

Descrição gerada automaticamente

Figura - simpleType generoType

Por fim criamos o simpleType “especialidadeType”, que é restringido para ser do tipo string e ter um limite máximo de 100 caracteres e um limite mínimo de 1 caracter.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - simpleType especialidadeType

## RelatorioTransferencias.xml

Este é o modelo do xml obtido após a realização do request http ao endpoint “getRelatorioTransferencias” e tratamento da reposta obtida através da função Xquery por nós realizada. A estrutura xml das Transferências, deste exemplo está divida em três elementos, sendo eles:

* “hospitais”;
* “lista\_transferencias”;
* “resumo\_mensal”.

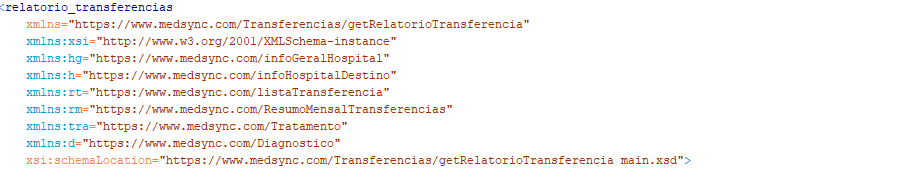


Figura – Namespaces RelatorioTransferencias.xml

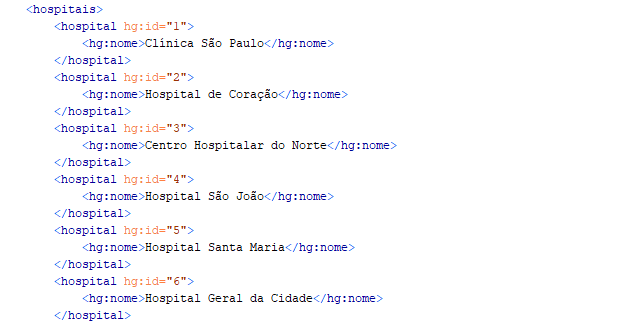
Como dá para observar na imagem 35, começamos por definir as namespace necessárias, para a criação do xml, como por exemplo a namespace “https://www.medsync.com/listaTransferencia” para os dados das transferências (xmlns:rt) . Para além disso, associamos o documento a uma schema XML externa, a “main.xsd”, com o uso do “xsi:schemaLocation”.

Figura – Elemento Hospitais

Na figura 36 dá para observar o elemento “hospitais”, que é o 1º elemento deste xml e armazena a informação de todos os hospitais (id que é um atributo e nome), de todos os hospitais que realizaram transferências no ano e mês em que se deseja fazer a pesquisa.

Nas imagens abaixo dá para observar o elemento “lista\_transferencias”, esse elemento é capaz de armazenar todos as transferências do relatório, onde os dados de cada transferência são armazenados no elemento “transferência”. O elemento “lista\_transferencias” é capaz de armazenar 0 ou infinitos elementos de transferência, dentro do mesmo. Já o elemento transferência é composto pelo atributo “id” e pelos seguintes elementos:

* cod\_paciente;
* data\_transferencia;
* motivo;
* tipo\_transferencia;
* relatorios\_medicos (que pode conter 0 ou infinitos relatórios médicos), que contêm para além do atributo “id” os seguintes elementos:
  + id\_profissional;
  + especialidade;
  + data\_atendimento;
  + diagnosticos, que é um array do elemento diagnostico, que por sua vez para alem do atributo “Codigo\_CID10” possui os seguintes elementos:
    - tipo\_diagnostico;
    - descricao;
  + tratamentos, que é um array do elemento tratamento, que por sua vez para alem do atributo “cod\_tratamento” possui o seguinte elemento:
    - tipo;



Figura – Exemplo do elemento lista\_transferencias

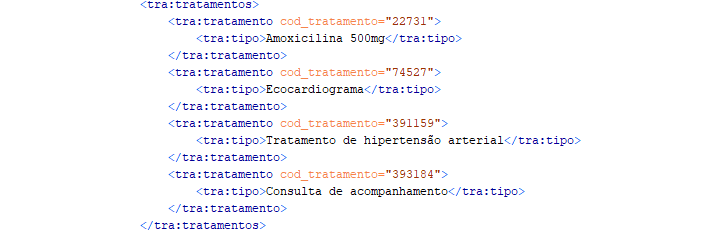


Figura – Exemplo do elemento tratamentos

Na imagem abaixo é possível observar um trecho 3º e último elemento deste xml, o elemento “resumo\_mensal” este elemento contém 4 elementos, sendo eles:

* tot\_transferencias;
* motivos, este elemento contém o número de transferências realizadas por motivos, é um array do elemento motivo, onde cada elemento possui os seguintes elementos:
  + nome;
  + tot\_transferencias;
* tipos, que é um array do elemento tipo, que por sua vez possui os seguintes elementos:
  + nome;
  + tot\_transferencias;
* hospitais, que é um array do elemento hospital, que por sua vez possui os seguintes elementos:
  + nome;
  + tot\_transferencias;

Figura – Exemplo do elemento resumo\_mensal

### main.xsd

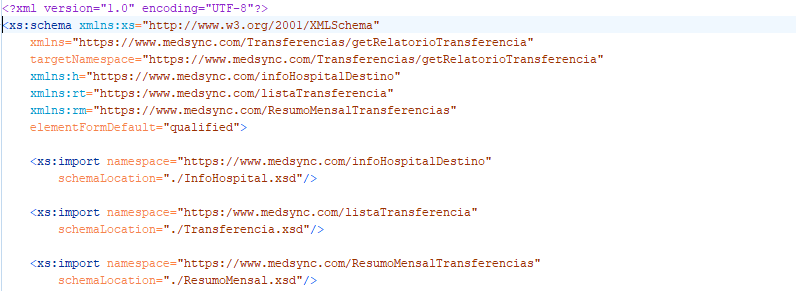
 Aqui falaremos da arquitetura do main.xsd, que é um xsd que importa múltiplos documentos xsd. Como dá para observar na imagem abaixo, no main.xsd começamos por declarar o namespace e importar ficheiros .xsd que foram utilizados.

Figura – Declaração dos namespaces do main.xsd

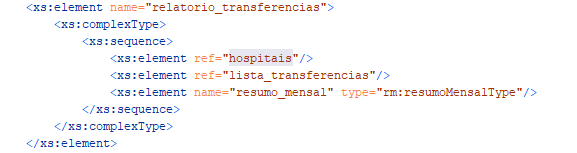
 Depois criamos o elemento “relatorio\_transferencias”, como dá para observar na imagem abaixo, que é o elemento principal deste xsd. O mesmo possui um complexType e dentro do mesmo é definida a sequência dos seus elementos filhos. Onde vai haver um elemento “hospitais”, que está a ser referenciado, o elemento “lista\_transferencias”, que também é referenciado e o “resumo\_mensal”, que é do tipo “rm:resumoMensalType”.

Figura - Declaração do elemento relatorio\_transferencias no esquema main.xsd

O elemento que dá para observar na imagem abaixo, é o “lista\_transferencias”, o mesmo elemento referenciado na imagem acima, só que nesta imagem a sua estrutura é definida, definimos-lho como um complexType, com uma sequência de um elemento, o elemento “transferencia”, que é do tipo “rt:transferenciaType” e pode ocorrer no mínimo 1 vezes e no máximo infinitas vezes.

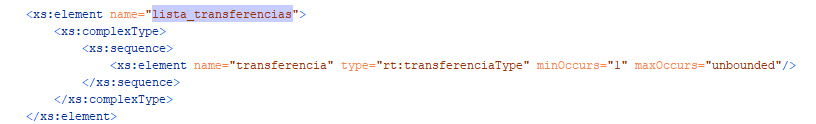


Figura - Declaração do elemento lista\_transferencias no esquema main.xsd.

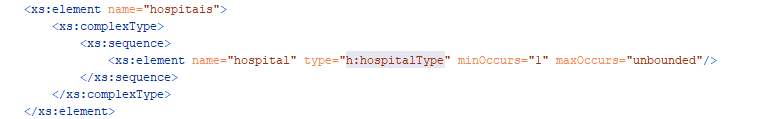
 Por fim temos o elemento “hospitais”, que tem um tipo complexo, com uma sequência de 1 elemento, com o elemento “hospital”, que é do tipo “h:hospitalType” e esse elemento pode ocorrer no mínimo 1 e no máximo infinitas vezes.

Figura - Declaração do elemento hospitais no esquema main.xsd

### InfoHospitalDestino.xsd

Neste tópico falaremos do arquivo InfoHospital.xsd, que utiliza também o arquivo infoGeralHospital.xsd. Como dá para ver na imagem abaixo, começamos por definir o targetNamespace e o xmlns deste xsd e inicializar o namespace “xmlns:h="https://www.medsync.com/infoGeralHospital"”.

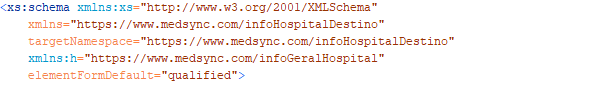
 Depois definimos o complexType “hospitalType”, que possui uma sequência com somente um elemento, que é uma referência para o elemento do xsd “InfoGeralHospital.xsd”, sendo ele o “h:nome”. O complexType ainda possui um atributo, sendo ele o “h:id”, que também pertence ao xsd “InfoGeralHospital.xsd” e que é de uso obrigatório.

Figura - Declaração do namespace e importação do xsd necessário

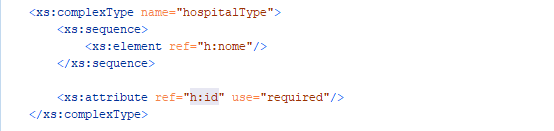


Figura - Definifção do complexType hospitalType

### Transferencias.xsd

Transferencias.xsd é um arquivo que define a estrutura do elemento que possui nenhuma ou infinitas transferências. Como é possível observar na imagem 46, inicialmente é declarado o namespace e é feita a importação de outros ficheiros xsd necessários, que neste caso são Tratamento.xsd e Diagnostico.xsd.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Declaração do namespace e importação de outros xsd necessários.

É possível observar na figura 47 as regras para a construção do elemento “transferenciaType”, que possui o atributo id do tipo “idType”, de uso obrigatório, também contem uma sequência, que é composta pelos seguintes elementos:

* cod\_paciente, do tipo idType;
* data\_transferencia, do tipo xs:dateTime;
* motivo, do tipo motivoType;
* tipo\_transferencia, do tipo tipoTransferenciaType;
* Uma imagem com texto, captura de ecrã, Tipo de letra

  Descrição gerada automaticamenterelatórios\_medicos, que é uma referência, e tem uma ocorrência mínima nula e máxima igual a um.

Figura - Regras da construção do elemento transferenciaType

Na figura 48, é definido o simpletype “tipoTransferenciaType” que restringe os valores possíveis para strings específicas. Dentro desta restrição, são especificados dois valores enumerados possíveis:

* Urgente: Representa transferências de caráter emergencial;
* Eletiva: Representa transferências planejadas ou programadas.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura – Definição do simpleType tipoTransferenciaType

Como é possível observar na figura 49, é definido o simpletype “idType” que restringe os valores possíveis para números inteiros positivos.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Definição do simpleType idType

Na Figura 50, é definido o simpleType “motivoType” que restringe os valores possíveis pra uma string com no mínimo 1 e no máximo 250 caracteres.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Definição do simpleType motivoType

Na figura 51 é possível observar as regras de construção do elemento relatorios\_medicos, que é composto por uma sequência do elemento referenciado relatorio\_medico, que tem a ocorrência mínima nula e a máxima é indefinida.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Regras da construção do elemento relatorios\_medicos

Na figura 51 é possível observar as regras de construção do elemento relatorio\_medicos, que é composto pelo atributo id do tipo idType de uso obrigatório e por uma sequência dos seguintes elementos:

* Id\_profissional, do tipo idType;
* especialidade, do tipo especialidadeType;
* data\_atendimento, que é uma referência, e tem uma ocorrência mínima nula e maxima igual a um;
* tratamentos, que é uma referência, e tem uma ocorrência mínima nula e máxima igual a um.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção do elemento relatorio\_medicos

Na figura 53 é definido o simpletype “especialidadeType” que restringe os valores possíveis para uma string de no mínimo 1 e máximo 250 caracteresUma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Definição do simpleType especialidadeType

### ResumoMensal.xsd

Definição da arquitetura do ResumoMensal.xsd, que é um xsd que importa múltiplos documentos xsd. Como dá para observar na imagem abaixo, no ResumoMensal.xsd começamos por declarar o namespace e importar os ficheiros .xsd que foram utilizados.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Declaração do namespace e importação dos xsd necessários

Na figura 55 é possível observar as regras da construção do elemento “tot-transferencias” e do elemento complexType resumoMensalType, onde o elemento “tot-transferencias” é do tipo “tot\_type”, e o “resumoMensalType” é composto de uma sequência de referências dos elementos:

* tot-transferencias;
* motivos;
* hospitais.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção do elemento tot-transferencias e do elemento complexType resumoMensalType

Na figura 56 é possível observar as regras da construção dos simpleTypes “tot\_type”, “nomeType”, “idType”, que possuem a seguinte estrutura:

* tot\_Type, restringido para somente possuir valores inteiros com o valor mínimo igual a 0;
* nomeType, restringido para somente possuir strings com o mínimo de caracteres igual a 1 e o máximo de caracteres 150;
* idType, restringido para somente possuir valores inteiros com o valor mínimo igual a 0.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção dos simpleTypes tot\_type, nomeType, idType

Na figura 57 estão descritas as regras da construção dos elementos motivos e motivo, que são as seguintes:

* motivo, que é composto por um complexType, que por sua vez possui uma sequência dos seguintes elementos:
  + nome, do tipo nomeType;
  + tot\_transferencias, que é uma referência.
* motivos, que é composto por um complexType, que por sua vez possui uma sequência de motivo, com uma ocorrência mínima igual a um e ocorrência máxima infinito;

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção dos elementos motivos e motivo

Na figura 58 estão descritas as regras da construção dos elementos tipos e tipo, que são as seguintes:

* tipo, que é composto por um complexType, que por sua vez possui uma sequência dos seguintes elementos:
  + nome, do tipo tipoType;
  + tot\_transferencias, que é uma referência.
* tipos, que é composto por um complexType, que por sua vez possui uma sequência de tipo, com uma ocorrência mínima igual a um e ocorrência máxima igual a dois.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção dos elementos tipos e tipo

Na figura 59 estão descritas as regras da construção dos elementos hospitais e hospital, que são as seguintes:

* hospital, que é composto por um complexType, que por sua vez possui uma sequência dos seguintes elementos:
  + nome, que é uma referência para nome;
  + tot\_transferencias, que é uma referência.
* hospitais, que é composto por um complexType, que por sua vez possui uma sequência de hospital, com uma ocorrência mínima igual a um e ocorrência máxima infinita,

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Regras da construção dos elementos tipos e tipo

## XSD Gerais

Neste tópico iremos falar de xsd gerias que criamos, ou seja, xsd que estão a ser usados em ambos os xml, sendo que criamos um total de 3 xsd gerais onde para falar de cada um deles dividimos este tópico em 3 sub-tópicos, sendo eles:

* InfoGeralHospital.xsd;
* Diagnosticos.xsd;
* Tratamentos.xsd.

### InfoGeralHospital.xsd

Começamos por definir o prolog (primeira linha de qualquer documento XML - neste caso, estamos na presença de um documento XSD, no entanto, este tipo de documento também utiliza linguagem XML na sua escrita). Este descreve a versão (neste caso, 1.0) e o tipo de codificação utilizado (neste caso, UTF-8).

O elemento *schema* é o *root* de qualquer documento XSD e contém a declaração do namespace XSD. *xmlns* e *targetNamespace* – Descrevem o namespace por defeito, indicando que todos os elementos utilizados neste documento estão declarados no namespace “https://www.medsync.com/infoGeralHospital. *elementFormDefault* – indica que todos os elementos neste documento estão associados ao *target* *namespace*.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - schema InfoGeralHospital.xsd

Aqui definimos os elementos “nome” e “morada”, que referenciam os tipos simples “nomeType” e “moradaType”.

O primeiro tipo simples restringe o nome do hospital a uma string (<xs:restriction base=”xs:string”>) comprimida entre 1 e 100 caracteres inclusive (<xs:minLength value=”1”/> e <xs:maxLength value=”100”/>). O segundo tipo simples restringe a morada do hospital igualmente a uma string (<xs:restriction base=”xs:string”>), porém comprimida entre 1 e 150 caracteres inclusive (<xs:minLength value=”1”/> e <xs:maxLength value=”150”/>).

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - elementos InforGeralHospital.xsd

Aqui definimos o atributo “id”. Existe uma restrição que solicita um valor inteiro (xs:restriction base=”xs:int”) e que impede que o valor do atributo seja igual ou inferior a 0 (xs:minExclusive value=”0”).

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

Figura - atributo id InfoGeralHospital.xsd

### Diagnosticos.xsd

Começamos por definir o prolog (primeira linha de qualquer documento XML - neste caso, estamos na presença de um documento XSD, no entanto, este tipo de documento também utiliza linguagem XML na sua escrita). Este descreve a versão (neste caso, 1.0) e o tipo de codificação utilizado (neste caso, UTF-8).

O elemento *schema* é o *root* de qualquer documento XSD e contém a declaração do namespace XSD. *xmlns* e *targetNamespace* – Descrevem o namespace por defeito, indicando que todos os elementos utilizados neste documento estão declarados no namespace “https://www.medsync.com/Diagnostico. *elementFormDefault* – indica que todos os elementos neste documento estão associados ao *target* *namespace*.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - schema Diagnosticos.xsd

Aqui definimos os tipos simples “CID10Type”, “tipoDiagType” e “descricaoType”. O primeiro restringe o código do diagnóstico a uma string (<xs:restriction base=”xs:string”>) com, pelo menos, 1 caractere (<xs:minLength value=”1”/>). O segundo restringe o tipo do diagnóstico igualmente a uma string (<xs:restriction base=”xs:string”>), com valor predefinido: “Principal” (<xs:enumeration value=”Principal”/>) ou “Secundario” (<xs:enumeration value=”Secundario”/>). O terceiro restringe a descrição do diagnóstico também a uma string (<xs:restriction base=”xs:string”>), limitada a 250 caracteres (<xs:maxLength value=”250”/>).

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - SimpleTypes Diagnosticos.xsd

Aqui definimos os elementos “diagnosticos” e “diagnostico”. O primeiro referencia o segundo (ref=”diagnostico”), podendo o elemento “diagnostico” ocorrer infinitamente (maxOccurs=”unbounded”). O segundo tem um atributo “Codigo\_CID10” (que referencia o tipo simples “CID10Type”) obrigatório (use=”required”) e dois elementos *child* “tipo\_diagnostico” e “descricao”, que referenciam os tipos simples “tipoDiagType” e “descricaoType”, respetivamente.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - elementos Diagnosticos.xsd

### Tratamentos.xsd

Começamos por definir o prolog (primeira linha de qualquer documento XML - neste caso, estamos na presença de um documento XSD, no entanto, este tipo de documento também utiliza linguagem XML na sua escrita). Este descreve a versão (neste caso, 1.0) e o tipo de codificação utilizado (neste caso, UTF-8).

O elemento *schema* é o *root* de qualquer documento XSD e contém a declaração do namespace XSD. *xmlns* e *targetNamespace* – Descrevem o namespace por defeito, indicando que todos os elementos utilizados neste documento estão declarados no namespace “[https://www.XSDGeral.com/Tratamento](https://www.xsdgeral.com/Tratamento)”. *elementFormDefault* – indica que todos os elementos neste documento estão associados ao *target* *namespace*.

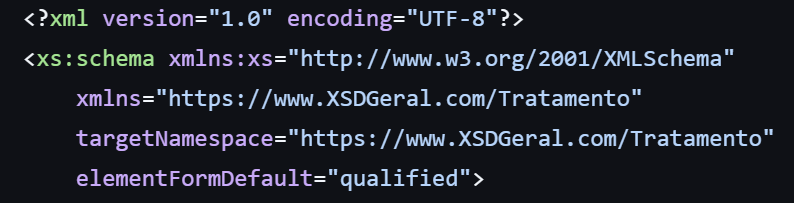


Figura - schema Tratamentos.xsd

Aqui definimos os tipos simples “tipoTratamentoType” e “idType” que vão ser referenciados pelo elemento “tratamento”. Existem duas restrições: a primeira, limita o comprimento da string de “tipoTratamentoType” a 250 caracteres; a segunda, não permite que sejam usados valores inteiros negativos para definir “idType”, possibilitando apenas valores maiores ou iguais a zero.

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

Figura - simpleTypes Tratamentos.xsd

Aqui definimos os elementos “tratamentos” e “tratamento”. O primeiro referencia o segundo (ref=”tratamento”), podendo o elemento “tratamento” ocorrer infinitamente (maxOccurs=”unbounded”). O segundo tem um atributo “cod\_tratamento” obrigatório (use=”required”) e um elemento *child* “tipo” que referencia o tipo simples “tipoTratamentoType”.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - elementos Tratamentos.xsd

# Mongo DB

O *MongoDB* é um sistema para a gestão de bases de dados, orientado a documentos, normalmente utilizado para armazenar grandes volumes de informação de maneira escalável e flexível. Em vez de armazenar os dados em tabelas, o *MongoDB* armazena-os em documentos, que são basicamente *JSON objects*. Além disso, é um sistema muito poderoso para consultas (inclui a criação de índices para melhorar a performance das consultas), e ainda possui um *framework* de agregação que permite processar e transformar os dados dentro da própria base de dados.

Tópicos a abordar:

* Modelação, conjunto das várias stages, utilizando o *framework* de agregação do *MongoDB*.
* Functions, funções utilizadas para a manipulação dos dados da base de dados *MongoDB*.

## Modelação

Neste tópico iremos abordar as técnicas de modelação que utilizamos para estrutura a base de dados fornecida para ficar com a melhor estrutura possível, para a realização das pesquisas no MongoDB.

Das coleções fornecidas, só tivemos a necessidade de modelar a coleção Registos Clínicos e a coleção Pacientes, onde entraremos mais em detalhe dessas modelações nos sub-tópicos deste tópico, onde o mesmo está divido em 2 sub-tópicos, sendo eles:

* Registos Clínicos, onde nesse tópico abordamos todas as modelações realizadas na coleção dos registos clínicos;
* Pacientes, onde nesse tópico abordamos todas as modelações realizadas na coleção dos pacientes.

### Registos Clínicos

A primeira alteração que fizemos aos registos clínicos foi agrupar todos os diagnósticos num array, pois reparamos que quando um registo clínico tinha mais do que um diagnostico associado, havia várias copias desse mesmo registo onde o que mudava era apenas os dados do diagnostico, então para evitar essa duplicação decidimos agrupar todos os diagnósticos num array de Diagnosticos, onde esse array teria todos os diagnósticos de um registo clínico específico. Para a realização dessa alteração começamos por ordenar pelo seu ID\_Atendimento e tipo de diagnósticos, como dá para observar na imagem abaixo.

Uma imagem com texto, software, Software de multimédia, captura de ecrã

Descrição gerada automaticamenteApós fazer a ordenação dos documentos realizamos o agrupamento dos dados, onde metemos como “\_id”, para o “$group” ser com base no id dos registos clínicos, também utilizamos o “$first” para guardar em cada um dos campos, “ID\_Atendimento”, “ID\_Paciente”, “ID\_Profissiona” e “Data\_Atendimento”, o valor de cada um desses campos do 1º atendimento do grupo. E no campo Diagnosticos, fizemos o “$push” de todos os dados dos diagnósticos, para criar o tal array de Diagnosticos.

Figura - Sort dos registos e tipo diagnostico

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - group dos Diagnosticos

A segunda alteração que fizemos não é muito importante, nós utilizamos um “$limt”, neste aggregate para trabalhar com um número menor de registos clínicos, onde utilizamos apenas 10000 registos clínicos, pode é possível, observar na imagem abaixo.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente A terceira alteração utilizamos o padrão de extended reference para embutir nos documentos a especialidade do medico de cada registo clínico, de forma a facilitar depois a pesquisa, pela especialidade do mesmo, sempre que fosse gerado o relatório dos registos clínicos. Onde para realizar essa alteração começamos por realizar um “$lookup” de forma a embutir, como dá para ver na imagem abaixo.

Figura - limit de registos clinicos

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Lookup dos profissionais

Depois de fazer este “$lookup”, utilizamos o “$unwind”, para transformar o array num objeto, uma vez que o array com os dados dos profissionais vai ter sempre um campo então em vez de essa informação, estar armazenada num array, é preferível ela ser um object, onde depois no “$project” metemos apenas para aparecer o campo especialidade, coisa que se estivesse a utilizar array não seria possível ou seria mais complicado. Na imagem abaixo apresento apenas o “$unwind” a parte do “$project”, irei apresentar mais abaixo no relatório, quando abordar a 3º alteração.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente A quarta e última alteração que fizemos foi embutir os tratamentos nos registos clínicos, de forma a facilitar a pesquisa pelos tratamentos prescritos e realizados associado ao relatório. Onde para o embutimento do mesmo utilizamos um “$lookup” para embutir os todos os dados dos tratamentos nos registos clínicos.

Figura - unwind Dados dos Profissionais

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente

Figura - loookup dos tratmanetos

Após realizar estas alterações realizamos um “$project” para modelar os o que vai ser mostrado nos documentos, como dá para ver na imagem abaixo. E como referido anteriormente é neste “$project” que metemos em prática o padrão extended reference, onde dos dados que embutimos dos profissionais vamos apenas mostrar a especialidade dele e não todos os dados do mesmo.

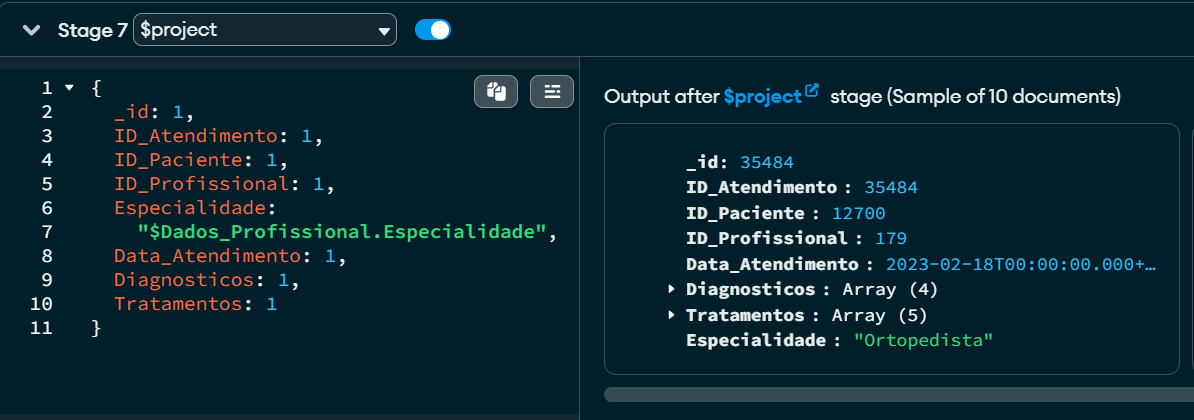
 Após estas alterações todas, ainda fizemos um “$out”, para guardar de forma definitiva todas estas alterações na coleção dos registos clínicos.

Figura - Project dos Registos Clinicos

Uma imagem com texto, captura de ecrã, software, multimédia

Descrição gerada automaticamente

Figura - out Registos Clinicos

### Pacientes

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente Nos pacientes nós decidimos embutir dentro do mesmo os registos clínicos otimizar a pesquisa pelo o Tipo de paciente e atualização do mesmo, porque como esse campo não existe nos pacientes, sempre que quisemos realizar uma pesquisa pelo o tipo ou precisaríamos de realizar um “$lookup“, pois sempre antes de mostrar o tipo paciente tem de ser verificado se é necessário realizar alguma atualização no campo, onde com os registos clínicos embutidos já não é necessário realizar esse “$lookup”, tornando assim tanto o processo de atualização do campo, como o de pesquisa mais rápido. Então para embutir esse campo utilizamos o “$lookup”, como dá para observar na imagem abaixo.

Figura - lookup dos registos clinicos nos pacientes

Outra modificação que fizemos nos pacientes foi o embutimento das transferências no mesmo, pois no relatório de transferência, para cada transferência realizada é necessário mostrar os relatório médicos/ registos clínicos, associados a essa transferência, então como ambos têm os campos “ID\_Paciente” e “ID\_Medico”, a nossa ideia, foi embutir também as transferências nos pacientes, para facilitar essa pesquisa, pois as transferências e registos clínicos com os mesmos pacientes iriam estar já no mesmo documento, facilitando assim essa pesquisa. Para esse embutimento também usamos um “$lookup”.Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - lookup das transferencias nos pacientes

Após fazer estes “$lookup”, optamos por fazer um “$project”, para mostrar apenas os dados, que nos interessavam, que eram todos dos pacientes, das transferências e dos registos, menos nas transferências e nos registos o campo “ID\_Paciente”, pois ele iria estar duplicado em todos os registos e transferências e como o próprio paciente já possuía esse campo era desnecessário ele estar a aparecer também nestes documentos.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente Como nos registos clínicos, para guardar de forma definitiva estes dados na coleção de pacientes utilizamos também o “$out”, como dá para ver na imagem abaixo.

Figura - project dos pacientes

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

### Indexes

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente Para otimizar as nossas pesquisas, e os lookups utilizados na modelação dos dados, decidimos meter indexes em alguns campos específicos, os mais utilizados nas pesquisas e os campos de ligação nos lookups, onde para os pacientes utilizamos indexes nos campos: “ID\_Paciente”, “Género”, “Data\_Nascimento” e “Email”, como dá para observar na imagem abaixo.

Figura - Indexes Pacientes

Já nos profissionais optamos por colocar apenas indexes no campo “ID\_Profissional”, como dá observar na imagem abaixo.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente Nos registos clínicos, optamos por meter indexes, nos campos: ” ID\_Atendimento”, “ID\_Paciente” e “ID\_Profissional”.

Figura - Indexes Profissionais

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente

Figura - Index Registos Clinicos

Para as duas coleções de tratamentos, a coleção “Tratamentos” e a coleção “TratamentosAtualizados”, metemos index no campo “ID\_Registro\_Clinico”

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Já para a coleção das transferências, colocamos um index no campo “ID\_Paciente”

Figura - Index Tratamentos

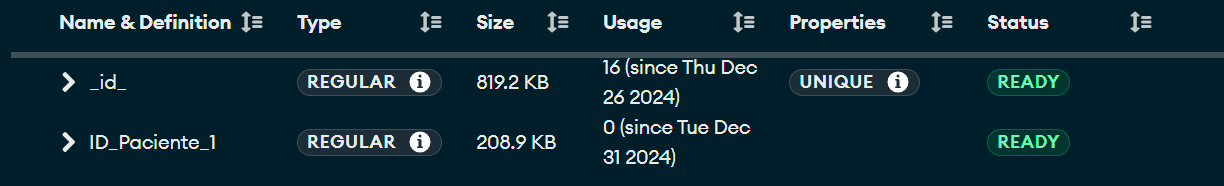


Figura - Index Transferências

## Functions

Neste tópico iremos falar de todas as funções que fizemos para a realização das pesquisas necessárias para fazer o relatório, e também dos puts, que realizamos sendo que de pesquisas realizamos as seguintes:

* getPacientes;
* getRegistosClinicos;
* getRelatorioRegistosClinicos;
* getResumoMensalRegistosClinicos;
* getResumoMensalRegistosClinicosPorGenero;
* getHospitaisTransferencia;
* getTransferencias;
* getRelatorioTransferencias;
* getResumoMensalTransferencias.

De puts, realizamos os seguintes:

* updateContactosPacientes;
* updateTipoPacientes;

Em todas as functions que abordamos todas iram receber como parâmetro os campos “mes” e “ano”, que representam o ano e mês que o utilizador quer consultar os dados, onde em todas as functions também colocamos duas verificações uma para ver se o mês colocado, que vai ser uma variável inteira, está entre o valor 1 até 12, que diz respeito há numeração dos meses, e outra verificação para verificar se o ano introduzido não é superior ao ano atual nem inferior a 1500, caso o mês e/ou ano seja valido será retornado o status 400 junto da mensagem de erro associado, como dá para observar na imagem abaixo.

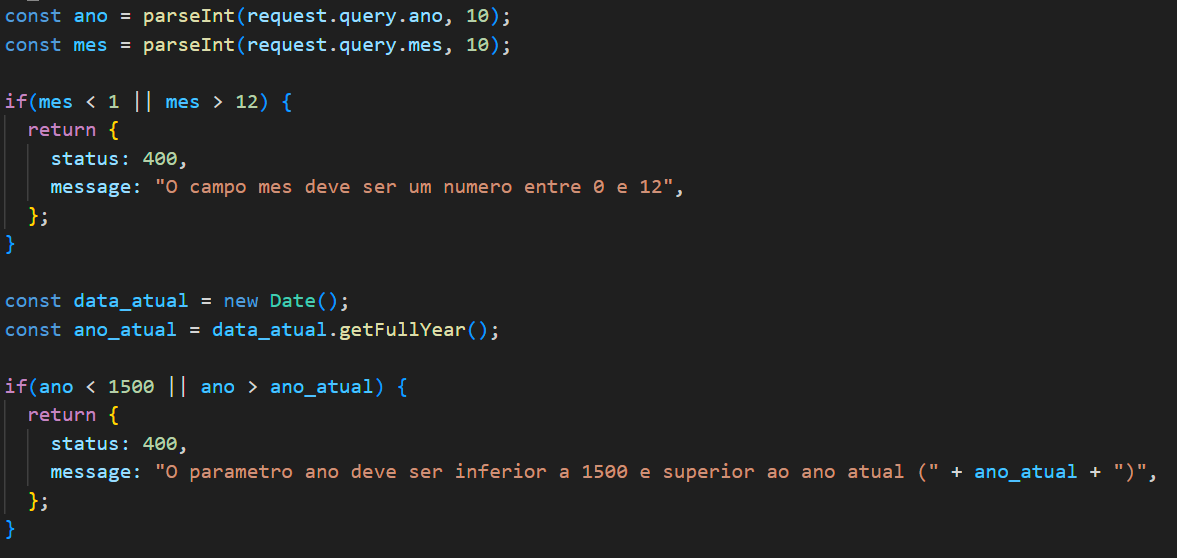


Figura - Verificação pelo ano e mes

Para além dessa verificação metemos um try catch em todas as functions, para que caso ocorre-se algum problema inesperado, fosse retornado o status 500, com a mesnagem de erro associada.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Erro status 500

### updateContactosPacientes

 Para atualizar o tipo de pacientes começamos por fazer conexão com a coleção dos pacientes.

Figura - Conexão coleção updateContactosPacientes

Ainda declaramos com constantes o primeiro dia do mês e o ultimo do dia, com base no mês recebido como parâmetro.



Figura - Constantes para a data minima e maxima do registo

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Após isso vamos atualizar o campo “Telefone” dos pacientes para “não fornecido”, em que a sua data de registo foi feita no mês e ano mandados como parâmetro e que o campo “Telefone” do paciente esteja vazio ou não exista, utilizando o updateMany.

Figura - codigo para atualizar telefone dos pacientes

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Depois atualizamos o campo “Email dos pacientes para “não fornecido”, em que a sua data de registo foi feita no mês e ano mandados como parâmetro e que o campo “Email”, esteja como “desconhecido”, utilizando o upateMany.

Figura - código para atualizar email dos pacientes

Caso seja possível atualizar os contactos dos pacientes é retornado o número de telefones e emails atualizados, juntamente com a mensagem “Atualização concluída” e é lançado o status 200.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Retorno atualizacao dos contactos

### updateTipoPaciente

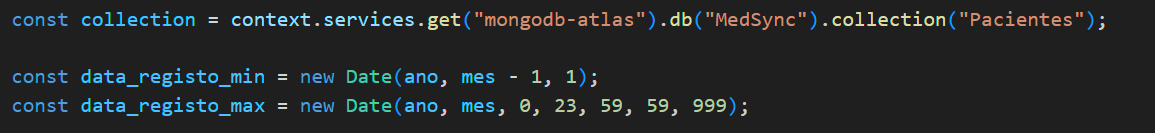
 Neste update tanto como no outro update, fazemos a conexão com a coleção dos Pacientes e ainda declaramos as mesmas variáveis, “data\_registo\_min” e “data\_registo\_max”.

Figura - Conexão e variáveis do updateTipoPaciente

Depois criamos a const pacientes, que vai guardar todos os pacientes que foram registados no mês recebido com parâmetro e inicializamos 3 array, que iram guardar os “ID\_Pacientes”, que precisam ser guardados para cada tipo.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - constante pacientes e arrays

Depois com um ciclo for percorremos cada um dos pacientes, e guardamos numa constante os dados, do paciente, na posição atual, do array de pacientes e inicializamos a variável que guarda qual vai ser o tipo daquele paciente, depois temos um if para verificar se aquele paciente possui pelo menos um registo clinico associado, se não tiver o tipo desse paciente vai ser novo, se tiver, vamos percorrer todos os seus registos clínicos e em cada registo clinico iremos percorrer todos os seus tratamentos, para ver se ele tem algum tratamento como não realizado, se ele tiver um tratamento por realizar ele será considerado como “Crónico”, se não como como “Regular”. Após descobrir o tipo do paciente, iremos guardar o id do paciente no array correspondente ao seu tipo, isso caso esse não seja o tipo atual do paciente.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura - Preenchimento dos arrays de tipo de paciente

Após percorrer todos os pacientes e descobrir qual vai ser o seu tipo, atualizamos o seu campo de Tipo de paciente o com o valor correspondido, isso se o tamanho de cada um dos array for superior a 0.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Atualização do campo de tipo de paciente

Caso a atualização do tipo seja realizada com sucesso é lançado o status 200, com a mensagem “Atualização concluída”, e é indicado o número de pacientes atualizados por tipo.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Retorno atualização dos pacientes

### getPacientes

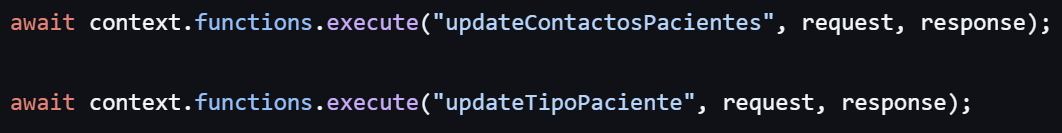
 Neste trecho de código são executadas duas funções: a primeira atualiza os contactos dos pacientes, enquanto a segunda atualiza o tipo de pacientes.

Figura - execução funções no getPacientes

Aqui acedemos à coleção “Pacientes” da base de dados “MedSync” e definimos o intervalo para o mês e ano a pesquisar. A seguir, filtramos os pacientes cuja data de registo está contida no intervalo definido anteriormente e para os pacientes encontrados mostramos apenas os campos necessários do mesmo (são os campos com o valor 1) e após realizar a pesquisa pelos pacientes retornamos os pacientes encontrados.

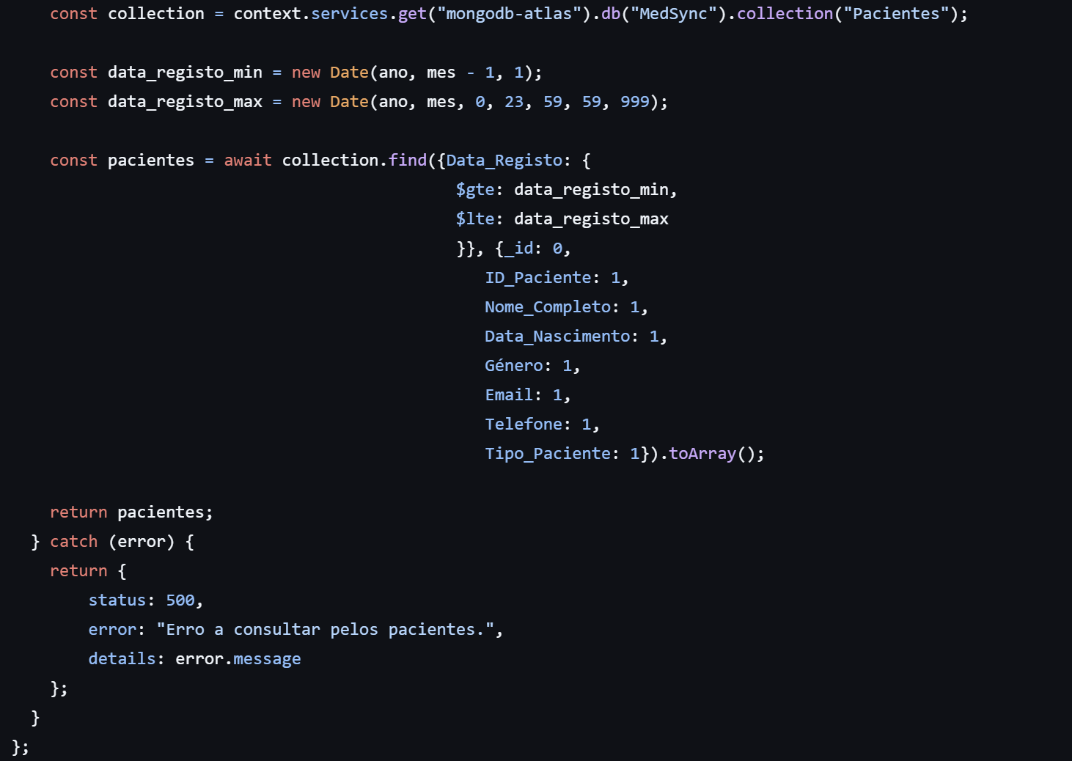


Figura - codigo do getPacientes

### getRegistosClinicos

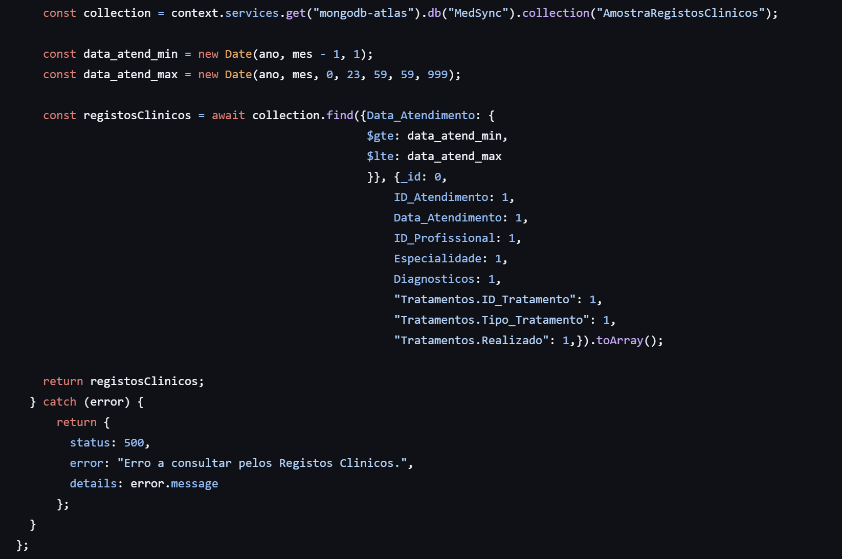
 Aqui acedemos à coleção “AmostraRegistosClinicos” da base de dados "MedSync” e definimos o intervalo para o mês e ano a pesquisar. A seguir, filtramos os registos clínicos cuja data de registo está contida no intervalo definido anteriormente e retornamos os campos marcados com “1” (o campo “\_id” está marcado com “0”, por isso fica “escondido”).

Figura - codigo para pesquisar pelos registos clinicos

### getRelatorioRegistosClinicos

Definimos o relatório para o hospital com id igual a 1 e definimos também um nome e morada default para o mesmo, para alem do mês e ano, do relatório. Executamos as funções que vão “carregar” o conteúdo do relatório em si – pacientes, registos clínicos e resumo mensal dos registos clínicos. Por último, retornamos o relatório.



Figura - codigo do getRelatorioRegistosClinicos

### getResumoMensalRegistosClinicos

Neste get começamos por criar uma constante para cada género, que conterá com um campo para o género que indica o género, o campo tot\_atendimentos que guarda o total de atendimentos feitos, o tot\_tratamentos que guarda o total de tratamentos realizados e o array de faixas\_etarias que guarda todos os dados de uma faixa\_etaria.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - constantes do genero

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente Depois começamos um ciclo for, que vai até ao valor 12, para representar as idades, de 0 até 120 em intervalos de 10 em 10, declaramos também as constantes para a idade mínima e máxima, da faixa etária e ainda definimos variáveis, para o ano de nascimento mínimo e máximo de cada uma das faixas etárias, para alem de criar uma constante para guardar a representação das faixas etárias. Depois guardamos numa constante o resultado da função “getResumoMensalRegistosClinicosPorGenero”, depois caso a constante possua dados e o seu tamanho seja superior a 0, então guardamo-la dentro do array de faixas etárias de cada género. O processo de executar o método e guardar no array de faixas etárias é feito para cada um dos géneros.

Figura - processo de guardar as faixas etarias

No final do ciclo for percorremos cada um dos array de faixas etárias de cada género e atualizamos o campo tot\_tratamenos e tot\_atendimentos de cada um dos géneros.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Total de atendiemntos e tratamentos de cada género

De seguida criamos a constante Resumo Mensal com o os campos “Total\_Atendimentos” e um array de duas posições com os dados do género Masculino e do género feminino e no final para atualizar o seu valor de total de atendimentos somamos o total de atendimentos de cada género e depois retornamos a contante “ResumoMensal”.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Varaivel Resumo Mensal dos registos clinicos

### getResumoMensalRegistosClinicosPorGenero

Neste get começamos por inicializar as constantes necessárias para fazer o resumo mensal de uma faixa etária de um género específico, que são uma constante para o ano, mês, faixa etária, o ano mínimo do registo do paciente de uma faixa etária especifica, o ano máximo, a data de atendimento mínima e máximo e a data de nascimento mínimo e máximo, e guardamos a coleção de pacientes numa constante.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente Depois realizamos um agregate para ir buscar todos os dados relevantes para cada faixa etária, onde fazemos um primeiro “$match”, para ir buscar os pacientes de um determinado género, mandado como parâmetro, pelos registos clínicos que existam e pertençam ao mês e ano mandado como parâmetro e ainda pelos utilizadores que nasceram no mês e ano mandado como parâmetro, como dá para ver na imagem abaixo.

Figura - Variaveis do getResumoMensalRegistosClinicosPorGenero

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Figura - match faixa\_etaria

Depois fazemos um “$unwind” para desfazer o array de RegistosClinicos

Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

Figura - unwind Registos Clinicos getResumoMensalRegistosClinicosPorGenero

Depois com o “$set” todos os registos clínicos que não têm tratamentos com um array vazio de tratamentos.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - Set Aggreate getResumoMensalRegistosClinicosPorGenero

Depois com o $addFields, criamos um campo chamado “Total\_Tratamentos”, que vai guardar o total de Tratamentos realizados em cada registo clinico, e criamos um campo chamado “Total\_Cronico” que guardado o total de casos crónicos em cada registo Clinico.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - aggreate getResumoMensalRegistosClinicosPorGenro addFields

Depois fazemos um group para agrupar todos os registos clínicos pelo o campo da Especialidade do Médico, onde também indicamos o Género, em cada um dos documentos, e o total de atendimentos, que soma sempre 1 com base no numero de registos clínicos de cada especialidade, o campo “Total\_Tratamentos”, que guarda o total de tratamentos de cada especialidade e o campo “Total\_Cronicos” que guarda o total de casos crónicos de cada especialidade.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Group especialidades getResumoMensalRegistosClinicosPorGenero

Depois criamos o campo “Faixa\_Etaria”, que vai indicar a faixa etária atual e o campo “Especialidade” que indica a especialidade médica.

Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

Figura - addFiedls 2 getResumoMensalRegisotsClinicosPorGenero

Depois voltamos a fazer um “$group” para agrupar todos os documento pela a Faixa\_Etaria, onde mostramos o id, a faixa etária, total de atendimentos, tratamentos e casos crónicos e para a faixa etária em especifica criamos um array de Especialidades com o campo “Especialidade”, “Total\_Tratamentos” e “Total\_Cronicos”.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Figura - group 3 getResumoMensalRegistosClinicosPorGenero

Por fim fazemos um “$project” para mostrar todos os dados falados anteriormente, tirando o “\_id” e depois retornamos o resultado deste aggregate.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - retorno getResumoMensalRegistosClinicosPorGenero

### getHospitaisTransferencia

Aqui acedemos à coleção “Transferencias” e definimos o intervalo para o mês e ano a filtrar pela pipeline de agregação ($gte: dataInicio e $lte: dataFim). No segundo stage, projetamos todos os hospitais de destinos de transferências e depois utilizamos o “$group” para agrupar cada um dos hospitais pelo nome do mesmo, assim removendo possíveis duplicações de hospitais de destino, pois um hospital de destino pode aparecer em mais do que uma transferência, e garantimos que mostramos todos os hospitais de destino sem mostrar hospitais duplicados.

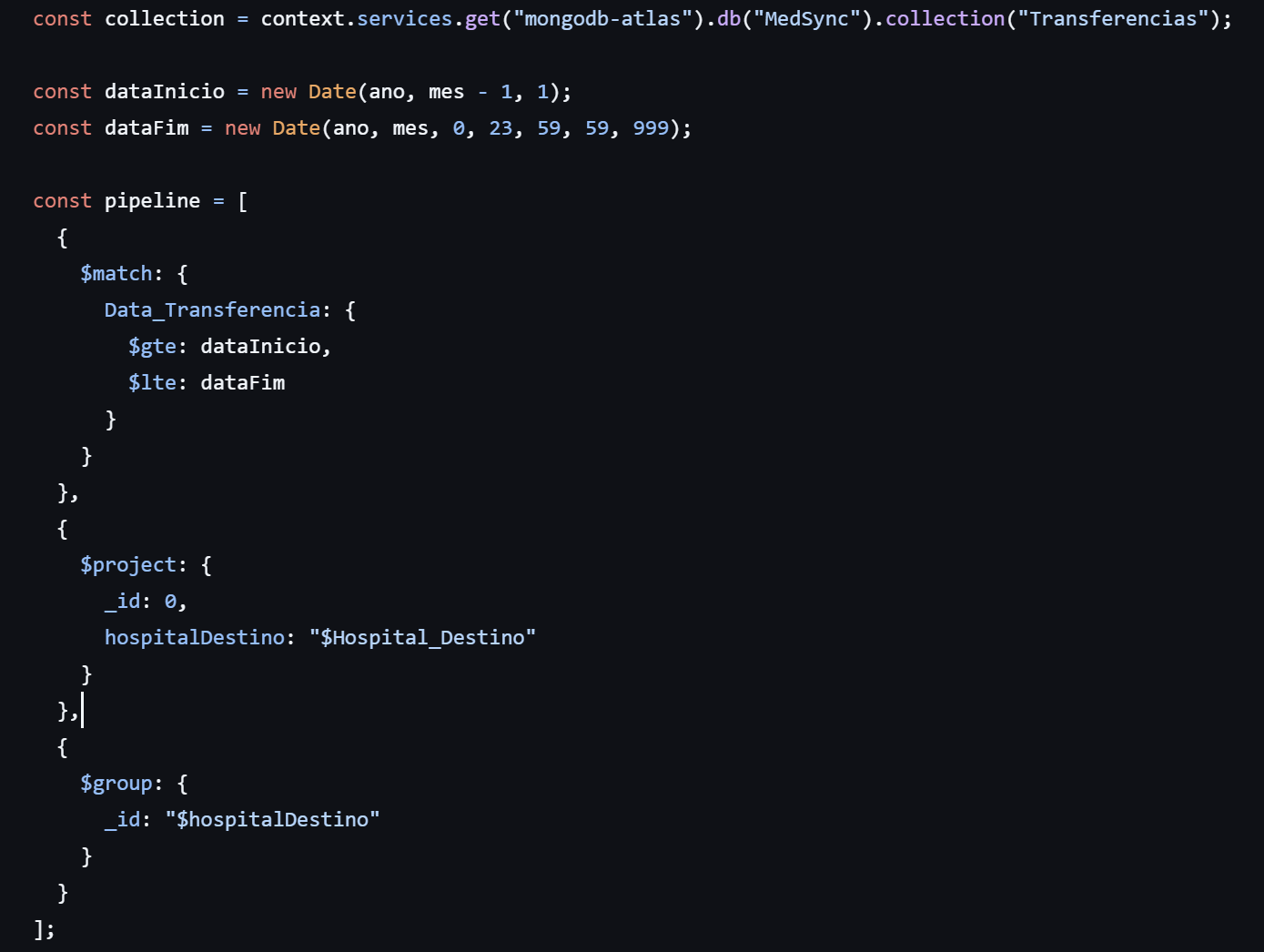


Figura - pipeline do getHospitaisTransferencia

Na imagem abaixo, é possível observar que criamos uma constante resultado, que executa um aggregate com a pipeline de agregação que criamos e que depois de executar realizamos um forEach, para adicionar um id, a cada um dos hospitais, de forma auto increment, e após a adição dos id, a cada um dos hospitais, retornados a variável result.

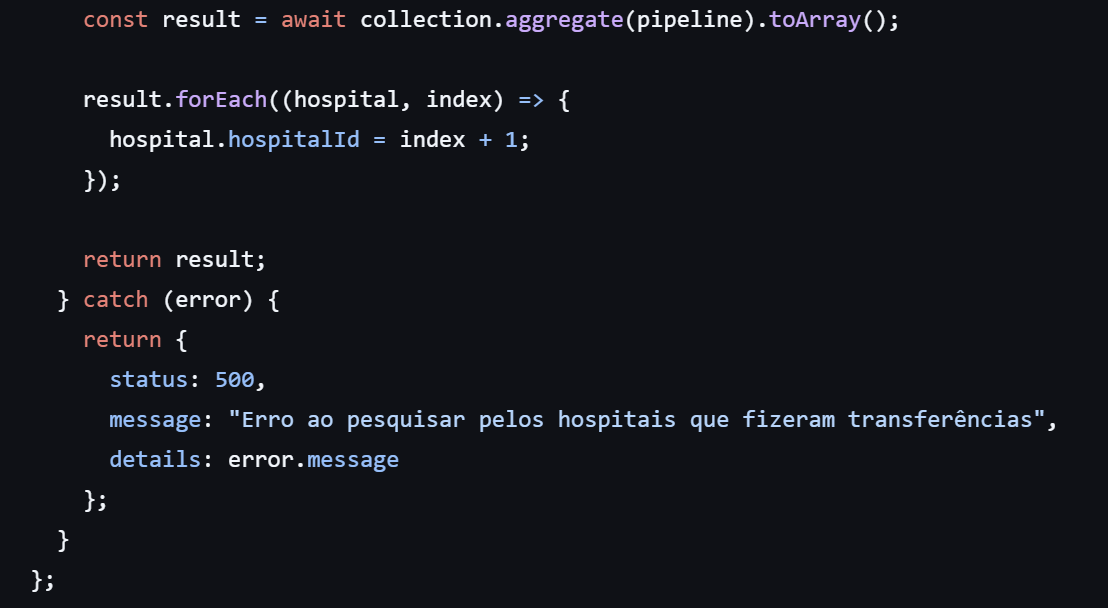


Figura - resultado pipeline getHospitaisTransferencia

### getTransferencias

Na figura a baixo é possível observar a definição da data de início e da data de fim da pesquisa desejada, e o inicio da pipeline, onde, dentre os dados dos pacientes realizamos os seguintes passos:

1. Filtragem para somente aparecer os pacientes que têm transferências
2. “Abrir” o array de transferências;
3. Filtrar novamente para só mostrar as transferências que foram realizadas antes da data de início e da data de fim;
4. Definição do primeiro project:

No primeiro $project, o objetivo é extrair e organizar os dados necessários. Aqui, os campos da coleção são projetados para criar uma estrutura inicial que inclua:

* Informações do paciente (ID\_Paciente);
* Detalhes da transferência:
  + ID\_Transferencia;
  + Data\_Transferencia;
  + Motivo;
  + Tipo\_Transferencia.
* Geração do campo RelatorioMedico com base nos registos clínicos (RegistoClinicos) anteriores à data da transferência. Nesse stage, utiliza-se:
  + $map: Para iterar sobre os registos clínicos;
  + $filter: Para incluir somente os registos cuja Data\_Atendimento seja anterior à Data\_Transferencia;
  + $ifNull: Para evitar erros no caso de ausência de dados. Para obter os dados relevantes e evitar que retorne dados nulos, evitando assim possíveis erros na conversão para xml (retorna os dados desejados, mas ainda não completamente filtrados ou formatados).

1. Definição do segundo Project:

No segundo $project, o foco é realizar ajustes finais e remoção de elementos indesejados. Especificamente:

* Refinamento do campo RelatorioMedico:
  + Garante que apenas registos médicos com tratamentos válidos sejam incluídos;
  + Usa um novo $map para iterar sobre RelatorioMedico e aplicar transformações específicas nos tratamentos:
    - Condição para verificar se existem tratamentos com o campo Realizado igual a "Sim";
    - Caso contrário, remove o campo Tratamentos com $REMOVE.
* Remoção condicional do RelatorioMedico vazio:
  + Se não houver registos médicos relevantes (ou seja, o tamanho da lista for 0), o campo RelatorioMedico é completamente excluído usando $cond e $REMOVE.
* Retorno final apenas dos campos relevantes.

1. Caso em qualquer fase teste percurso seja detetado um erro, retorna uma mensagem que diz “Erro ao pesquisar pelos dados da transferência” e fornece detalhes do erro.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura - Definifição da data de inicio e da data de fim, e o inicio da pipeline

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura – definição do primeiro $project

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura - Fim do primeiro project e inicio do segundo

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura - Fim do segundo project e fim da function

### getRelatorioTransferencias

Executamos as funções que vão “carregar” o conteúdo do relatório– hospitais de destino das transferências, transferências e o resumo mensal das transferências. Por último, retornamos o relatório.

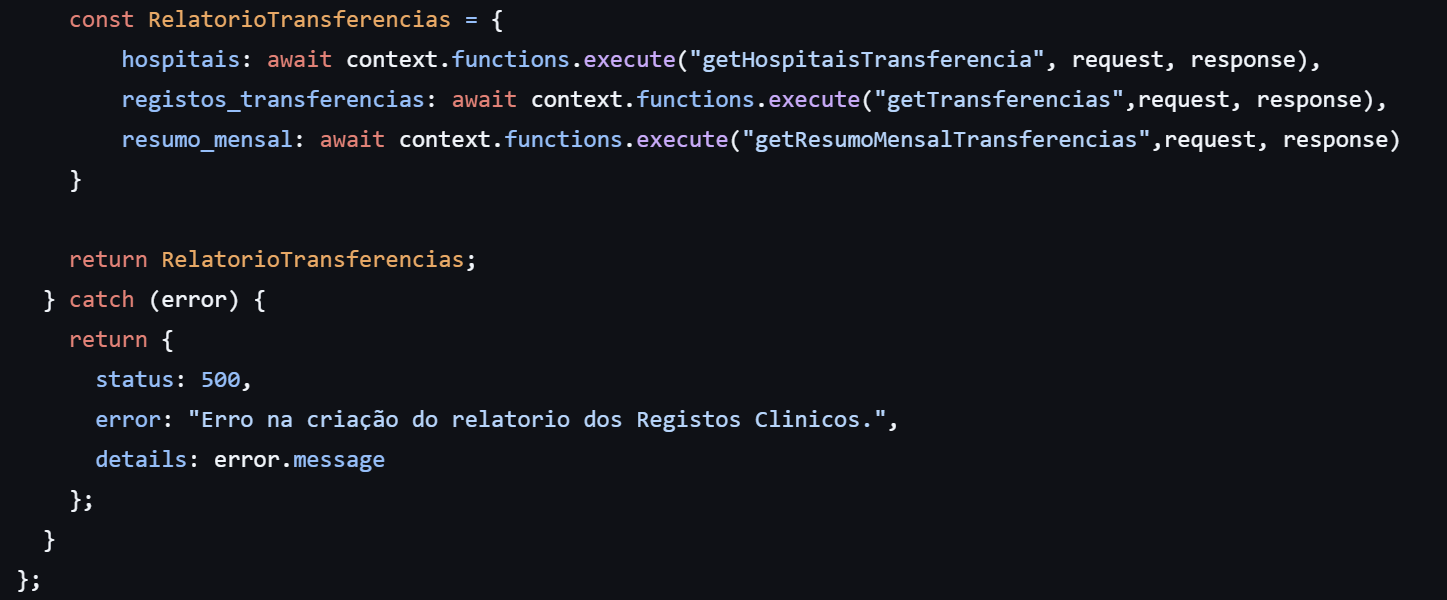


Figura - codigo getRelatorioTransferencias

### getResumoMensalTransferencia

Na figura a baixo é possível observar um trecho da function “getResumoMensalTransferencias”, que realiza a pesquisa dentro da collection transferências, e tem a função de retornar o os seguintes valores:

* Número total de transferências;
* Número de transferências ocorridas por motivos;
* Número de transferências por tipo;
* Número de transferências por hospital;

A function segue os seguintes passos:

1. Declaração da data de início e da data de fim da pesquisa através do input recebido (somente o mês e ano limite)
2. $match: Encontrar as transferências que combinam com as condições fornecidas, ou seja, têm a data de transferência maior que a data de início e menor que a data de fim
3. $facet: Utilizado para executar vários pipelines de agregação simultaneamente em uma única operação sem a necessidade de várias consultas ao banco de dados ou recorrer a mais que uma pipeline de agregação. É utilizado para calcular:

* O total de transferências (totalTransferencias).
* A distribuição por motivo (porMotivo).
* A distribuição por tipo de transferência (porTipo).
* A distribuição por hospital (porHospital).

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - Function getResumoMensalTransferencias

Como só fazemos a pesquisa para um mês e ano, só temos interesse em retornar o primeiro resultado do array que resulta da pipeline por nós desenvolvida. Já a mensagem de erro, é retornada sempre em conjunto com detalhes do erro.

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

Figura - Retorno do resultado e retorno de mensagem de erro caso necessário

# Postman

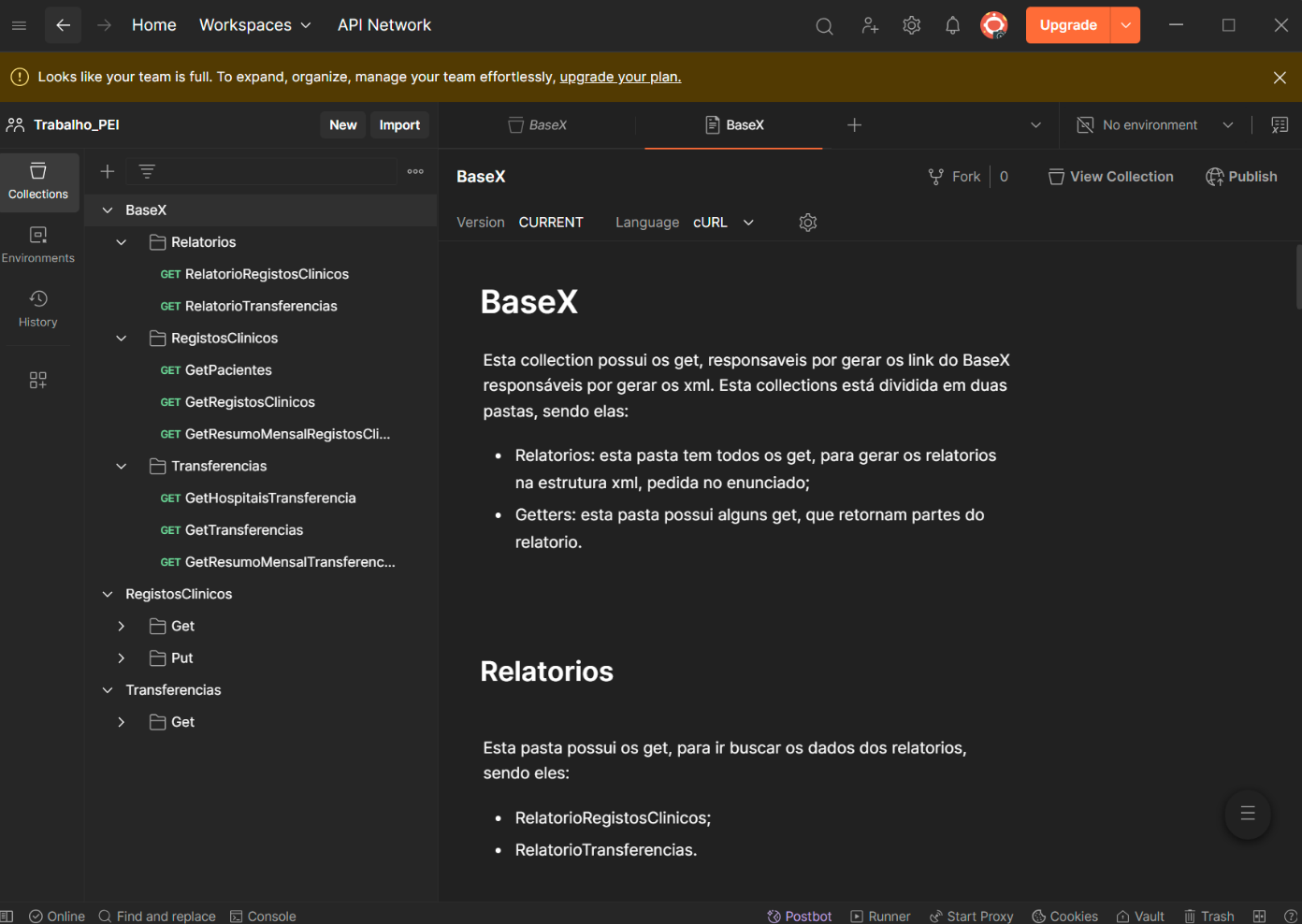
 Relativamente ao Postman foi a aplicação que nos permitiu testar as xquerys que desenvolvemos no BaseX usando requests. A documentação dos request criados está nos documentos gerados do postman.

Figura - Postman

# BaseX

BaseX é uma ferramenta completamente open source um mecanismo de banco de dados XML robusto e de alto desempenho e um processador de arquivos XQuery. Ele serve como uma excelente estrutura para a construção de aplicativos da Web complexos com uso intensivo de dados.

No âmbito deste trabalho o basex foi utilizado para fazer a conversão dos dados recebidos pelo request a API web em formato JSON para o formato XML, com recurso ao XQuery. Garante que todos os relatórios são gerados no formato XML padrão definido pelo MedSync (Presente nos .XSD). Garantimos assim que os relatórios cumpram as exigências do vocabulário CML comum.

Neste tópico iremos apresentar as XQuerys que realizamos para apresentar os dados dos relatórios em formato xml, sendo que dividimos cada uma em 2 tópicos, sendo eles:

* RelatorioRegistosClinicos.xq;
* RelatorioTransferencias.xq.

## RelatorioRegistosClinicos.xq

RelatorioRegistosClinicos.xq é um script XQuery desenvolvido para converter dados clínicos recebidos de um endpoint HTTP para um formato XML organizado. Esse XML é gerado de acordo com um conjunto de namespaces e esquemas específicos, representando informações hospitalares, pacientes, registros clínicos e um resumo mensal.Os passos para a ação ser realizada com sucesso são os seguintes:

1. Declaração dos namespace, declarados para organizar os elementos XML, garantindo que cada conjunto de dados siga o seu respetivo esquema.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Declaração dos namespace de RelatorioRegistosClinicos.xq

1. Configuração do Endpoint REST, para definir o endpoint acessível via HTTP.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - Configuração do Endpoint REST de RelatoriosRegistosClinicos.xq

1. Construção do URL de requisição, o URL é montada com os parâmetros fornecidos (mês e ano).

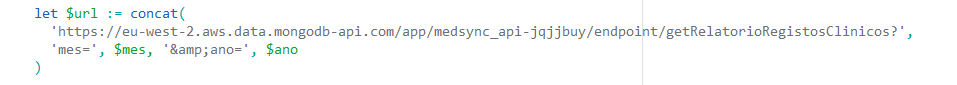


Figura - Construção do URL de requisição de RelatorioRegistosClinicos.xq

1. Envio de Requisição HTTP e Recepção de Dados, o módulo http:send-request é utilizado para enviar a requisição e obter os dados no formato JSON.

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

Figura - Requisição HTTP RelatorioRegistosClinicos.xq

1. Conversão dos Dados para XML
   * Elemento Raiz, organiza os dados e especifica o esquema XML usado para validação com o atributo xsi:schemaLocation.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Pequeno trecho do elemento principal RelatorioRegistosClinicos.xq

* + Elemento hospital, organiza as informações básicas do hospital.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - Elemento hospital RelatorioRegistosClinicos.xq

* + Elemento lista\_pacientes, contém a lista de pacientes com informações detalhadas.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura - Elemento lista\_pacientes RelatorioRegistosClinicos.xq

* + Elemento lista\_registos, representa os registos clínicos, incluindo diagnósticos e tratamentos.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura - Elemento lista\_ registos RelatorioRegistosClinicos.xq

* + Elemento resumo\_mensal, fornece um resumo categorizado por gênero e faixas etárias.

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Figura - Elemento resumo\_mensal RelatoriosRegistosClinicos.xq

## RelatorioTransferencias.xq

RelatorioTransferencias.xq é módulo XQuery que processa um request HTTP para obter e transformar dados relacionados a transferências hospitalares. Esses dados são estruturados em formato XML, com namespaces bem definidos para garantir compatibilidade e clareza. Os passos para a ação ser realizada com sucesso são os seguintes:

1. Declaração dos namespaces:
   * A definição dos namespaces ajuda a organizar os elementos XML gerados evitando conflitos entre nomes de elementos iguais que pertencem a diferentes arquivos XSD

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Declaração dos namespaces

1. Configuração do Endpoint REST
   * Essa seção define que o módulo XQuery será acessado via um endpoint REST com o método GET. Os parâmetros mes e ano são capturados da query string da requisição.

Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Figura - Declaração do Endpoint REST

1. Montagem do URL para Requisição HTTP
   * A URL é construída com os parâmetros mes e ano para buscar os dados correspondentes na API externa.

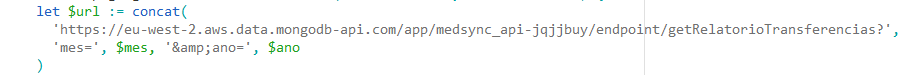


Figura - Construção do URL

1. Envio da Requisição HTTP e Processamento do Retorno
   * A função http:send-request realiza a requisição HTTP e obtém o conteúdo no formato JSON. Este é posteriormente transformado em XML para manipulação no XQuery.

Uma imagem com texto, Tipo de letra, escrita à mão, file

Descrição gerada automaticamente

Figura - Requisição HTTP

1. Geração da Estrutura XML:
   * Elemento Raiz, organiza os dados e especifica o esquema XML usado para validação com o atributo xsi:schemaLocation.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura - Pequeno excerto do elemento raiz

* + Lista de hospitais, cria uma lista de elementos hospital, extraindo o ID e o nome de cada hospital presente no relatório.

Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

Figura - Elemento hospitais

* + Lista de transferências, cada transferência é processada individualmente, com os campos código do paciente, data da transferência, motivo, tipo entre outros que é possível observar na figura a baixo sendo extraídos e estruturados.

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Figura - Elemento lista\_transferencias

* + Resumo mensal, resume as transferências por total, motivo, tipo e hospital, organizando os dados em elementos XML adequados.

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Figura - Elemento resumo\_mensal

# Conclusão

No final da realização deste trabalho prático vimos as nossas capacidades melhoradas para, através da utilização de tecnologias de estruturação documental, apoiar a criação, processamento, preservação e organização de coleções de documentos. Além disso, ficamos a compreender melhor como é que podemos armazenar e catalogar os dados de documentos estruturados, visando a gestão eficiente da informação. Então, vimos como é que os dados e os documentos podem ser trocados e reutilizados de forma eficaz.

# Referências

<https://moodle2.estg.ipp.pt/pluginfile.php/13089/mod_resource/content/2/APP3.pdf> (31/12/2024)

<https://moodle2.estg.ipp.pt/pluginfile.php/17403/mod_resource/content/0/APP8.pdf> (31/12/2024)

<https://moodle2.estg.ipp.pt/pluginfile.php/17873/mod_resource/content/1/APP9.pdf> (31/12/2024)

<https://moodle2.estg.ipp.pt/pluginfile.php/18872/mod_resource/content/1/APP11.pdf> (31/12/2024)

<https://www.mongodb.com/docs/manual/reference/operator/aggregation/facet/>

(01/01/2025)

<https://www.mongodb.com/pt-br/docs/manual/reference/operator/aggregation/map/>

(01/01/2025)

<https://basex.org> (01/01/2025)