

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

Wilkie Filho nº8230127

Francisco Xico nº8230148

Ano Letivo: 2024/2025

Índice

[**Introdução**](#_o8jv4bp9gtza) **4**

[**Estrutura XML**](#_wr15m7p5gadj) **5**

[RelatorioRegistosClinicos.xml](#_ojgm6lo8flx1) 6

[main.xsd](#_z69ehzsbevk4) 7

[InfoHospitalRegisto.xsd](#_p5u6vjn7qhaw) 8

[PacientesAtendidos.xsd](#_v4le5sm4toca) 9

[RegistoClinico.xsd](#_lxi96ms5nm3a) 10

[ResumoMensal.xsd](#_b43rgzje2tmn) 11

[Transferencias.XML](#_lrle4dpru5rf) 12

[main.xsd](#_rsr718nkkka3) 13

[InfoHospital.xsd](#_r4plhyxt9b8) 14

[Transferencias.xsd](#_2rpbhke5s7t7) 15

[ResumoMensal.xsd](#_x9lmc94p081b) 16

[ValoresGerais.xsd](#_q41d5tuekjmf) 17

[XSD Gerais](#_jdm99hcvsaij) 18

[InfoGeralHospital.xsd](#_phak8mcgw321) 19

[Diagnosticos.XSD](#_knfx8ut6azni) 20

[Tratamentos.xsd](#_9erb66p7mc7i) 21

[**Mongo DB**](#_7x67l9dld7dg) **22**

[Modelação](#_r1088wp8yfa4) 23

[Registos Clínicos](#_sf65dpqf3pbg) 24

[Pacientes](#_9enj7ows502g) 25

[Functions](#_sd3lnx5zu39f) 26

[getPacientes](#_g91nuuptinow) 27

[getRegistosClinicos](#_x0izeymtyw88) 28

[getRelatorioRegistosClinicos](#_9vi4jkmebued) 29

[getResumoMensalRegistosClinicos](#_kit3zp1kmoq9) 30

[getResumoMensalRegistosClinicosPorGenero](#_p7d48botvnyb) 31

[updateContactosPacientes](#_uf0m0gfnpezt) 32

[updateTipoPaciente](#_p8w5o5pmjfzi) 33

[getHospitaisTransferencia](#_7un9nsxbgpj2) 34

[getTransferencias](#_izl8g8ou82lh) 35

[getRelatorioTransferencias](#_sguvngkshr3s) 36

[getResumoMensalTransferencia](#_al3kdt4yerq1) 37

[**Postman**](#_y233h9ahd5) **38**

[**BaseX**](#_tedmemenxo6m) **39**

[RelatorioRegistosClinicos.xq](#_2jdykhnms7h3) 40

[RelatorioTransferencias](#_d8ebl6lkat4o) 41

[**Conclusão**](#_qkjt5wtg09d) **42**

Topicos feitos

Índice

Introdução – Por fazer

Estrututa XML - Por fazer

Ficheiros XML - Por fazer

Meter Todos os ficheiros - Por fazer

Ficheiros XSD - Por fazer

Meter texto para todos os xsd - Por fazer

MongoDB - Por fazer

Modelação- meter pritns e explicar as modelações e alterações que fizemos na BD

Functions - Falar de cada uma das functions

EndPoints - Mostrar os EndPoints criados (Talvez não seja necessario)

Postman - Por fazer (Talvez seja mais interresante apagar o Indice dos EndPoints e focar apenas neste, falar de cada link ou e falar da organização feita)

BaseX - Por fazer (Neste caso talvez só falar dos Principais os dos Relatorios os outros criados não são relevantes)

Conclusão

(Meter também automatica depois)

Indice de Imagens

# Introdução

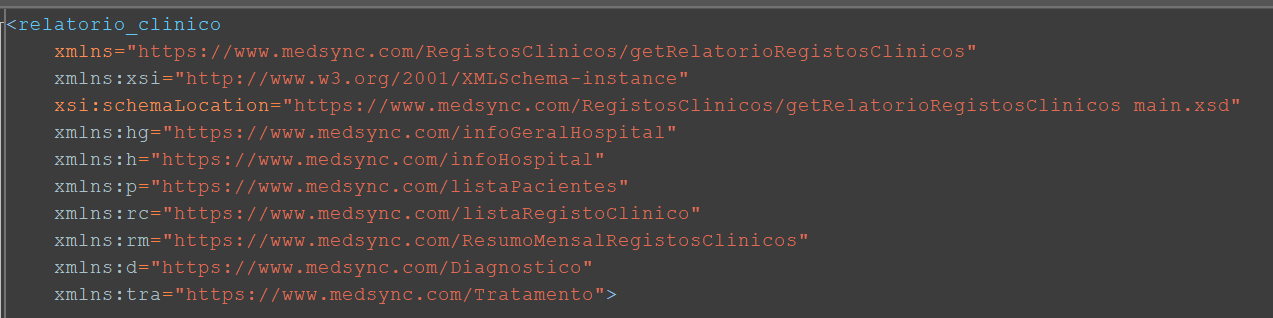
(Meter texto de Introdução)

# Estrutura XML

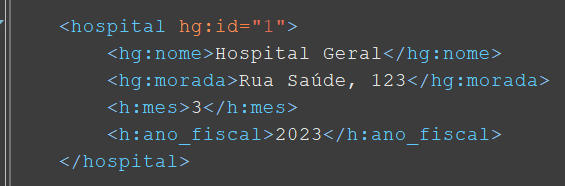
(Textinho de introdução ao topico, a explicar o que vai ser falado no mesmo e explicar a estrutura dele, meter também uma lista a dizer os subTopicos deste topico, mas sõ dos XML e o topico XSD Gerais)

## RelatorioRegistosClinicos.xml

A estrutura xml dos Relatórios dos Registos Clínicos, deste exemplo está divida em quatro elementos, sendo eles:

* “hospital”;
* “lista\_pacientes”;
* “lista\_registos”;
* “resumo\_mensal”.

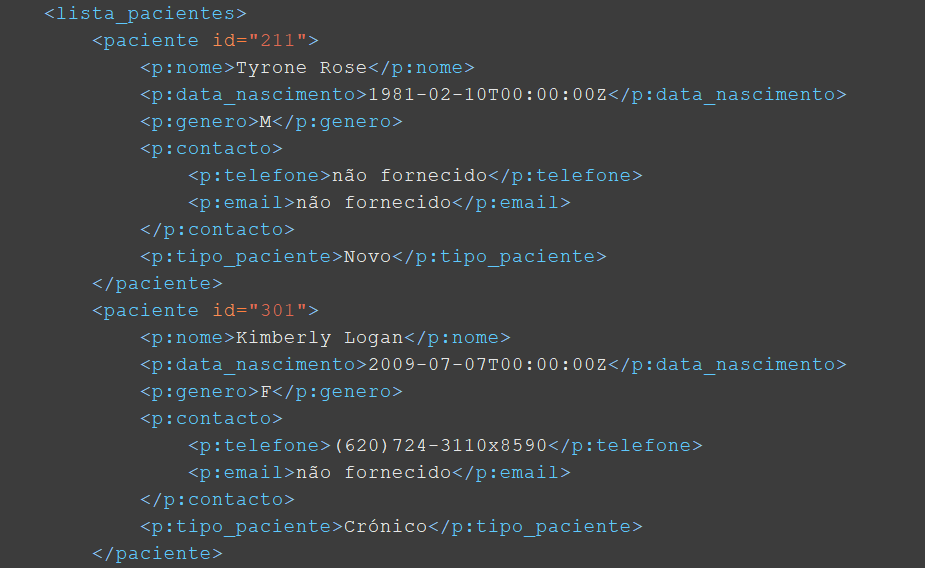
Como dá para observar na imagem acima, começamos por definir as namespace necessárias, para a criação do xml, como por exemplo a namespace “https://www.medsync.com/listaPacientes” para os dados dos pacientes (xmlns:p) . Para além disso associamos o documento a uma schema XML externa, a “main.xsd”, com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



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 (elemento h:ano\_fiscal) do relatório.

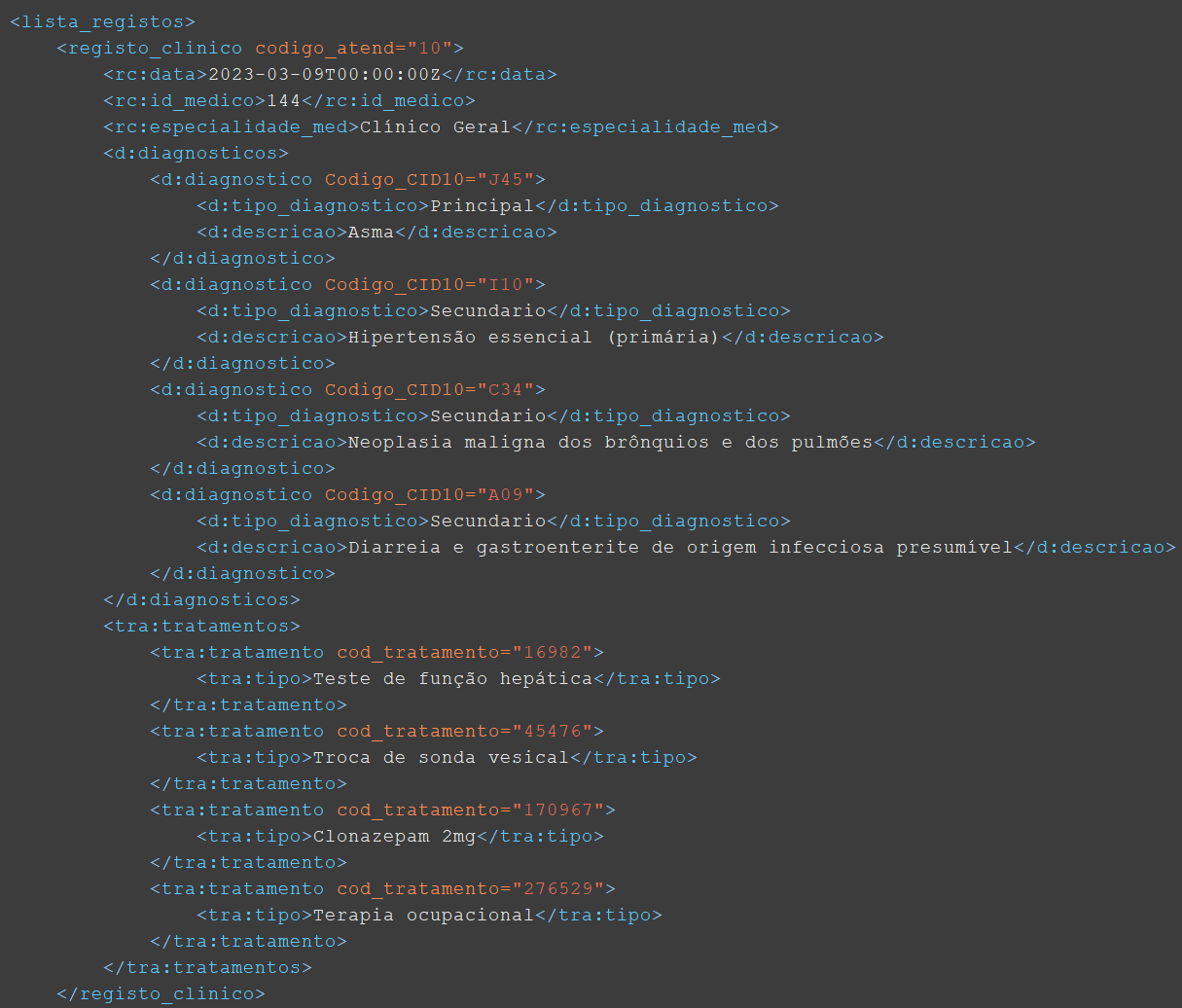
Na imagem abaixo dá para observar o elemento “lista\_pacientes”, esse elemento é capaz de armazenar todos os pacientes do relatório, onde os dados do paciente são armazenados no elemento “paciente”. O elemento “lista\_pacientes” é capaz de armazenar 0 ou infinitos elementos paciente, dentro do mesmo. 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.



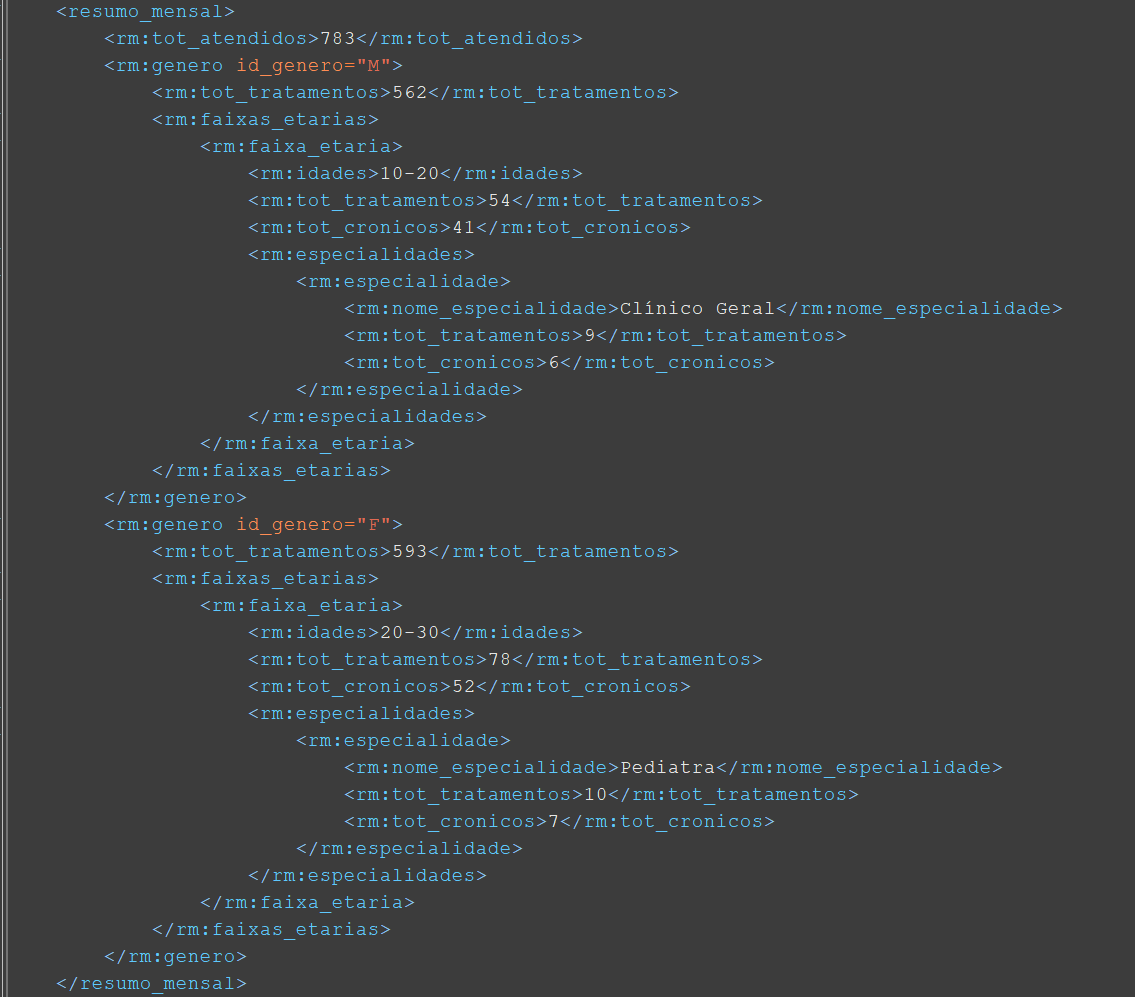
A imagem abaixo mostra o 3º elemento do XML, o elemento “lista\_registos”, que têm o mesmo funcionamento que o elemento “lista\_pacientes” o que muda entre os dois é que enquanto a “lista\_pacientes” mostra os dados dos pacientes este elemento mostra os dados dos registos clínicos. Os registos clínicos, 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.



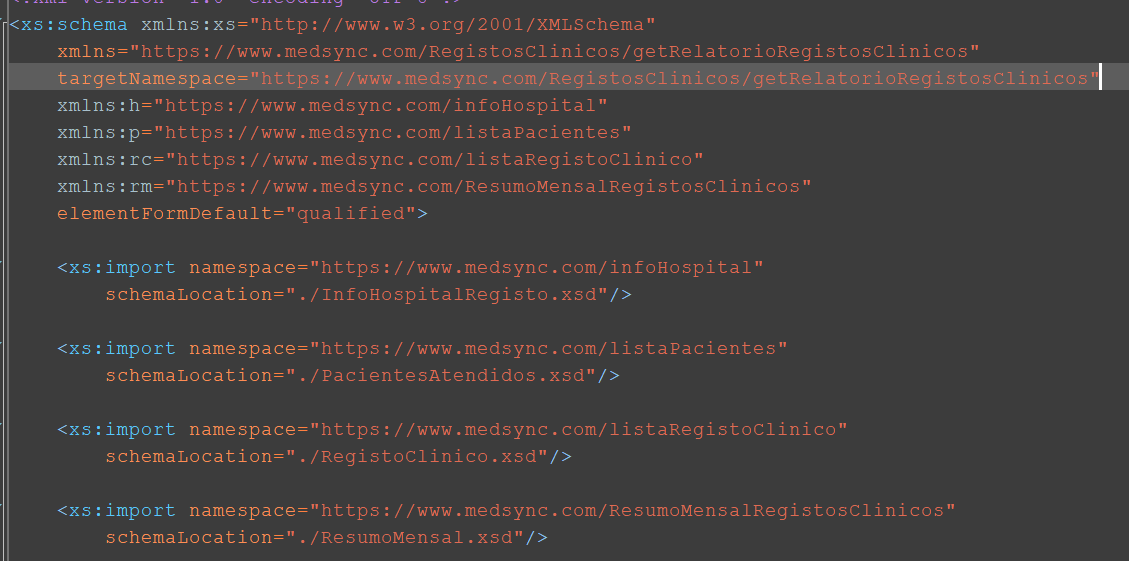
Na imagem abaixo é possível observar o 4º e último elemento deste xml, o elemento “resumo\_mensal” este elemento contém 2 elementos, sendo eles:

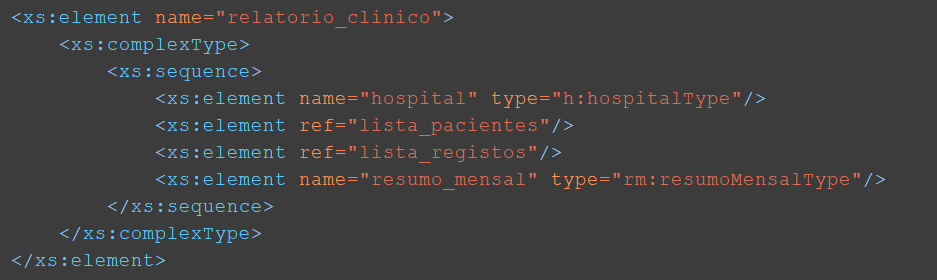
* tot\_atendimentos;
* genero, este elemento aparece obrigatoriamente 2 vezes, pois cada um contém a informações referente a cada genero (masculino e feminino), cada um contém o atributo id\_genero, e possui os seguinte elementos:
  + tot\_tratamentos;
  + faixas\_etaria, este elemento guarda todas as 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.



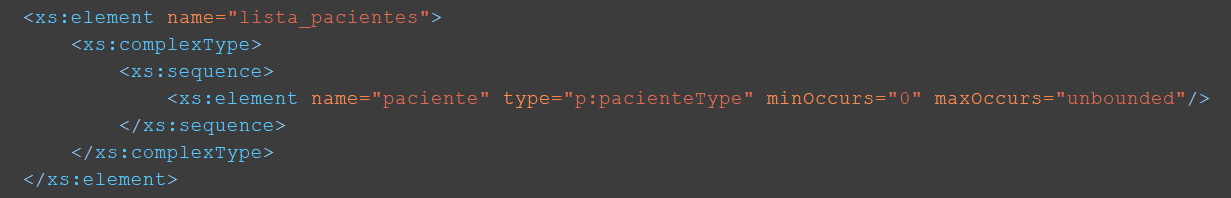
### main.xsd

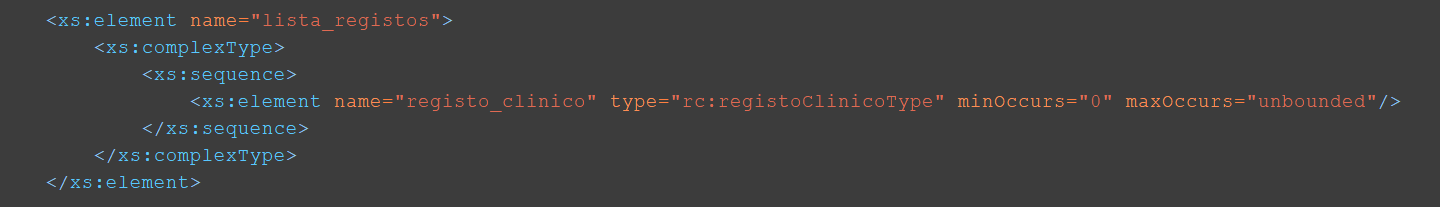
Como dá para observar na imagem abaixo, no main.xsd começamos por declarar os namespace e importar os mesmos.

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 a sequencia dos seus elementos filhos. Onde vai haver um elemento hospital do tipo “h:hospitalType”, o elemento “lista\_pacientes”, que está a ser referenciado, o elemento “lista\_registos”, que também é referenciado e o “resumo\_mensal”, que é do tipo “rm:resumoMensalType”.

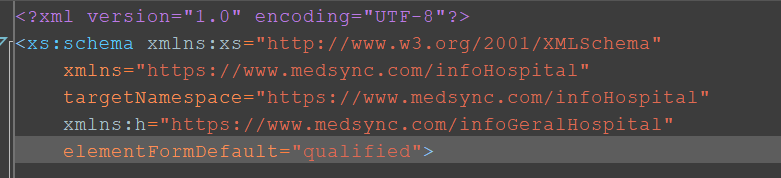


O elemento que dá para observar na imagem abaixo, é o “lista\_pacientes”, o mesmo elemento referencia na imagem acima, só que aqui nos definimos a estrutura dele, onde definimos-lo como um complexType, com uma sequencia de um elemento, o elemento “paciente”, que é do tipo “p:pacienteType” e pode ocorrer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

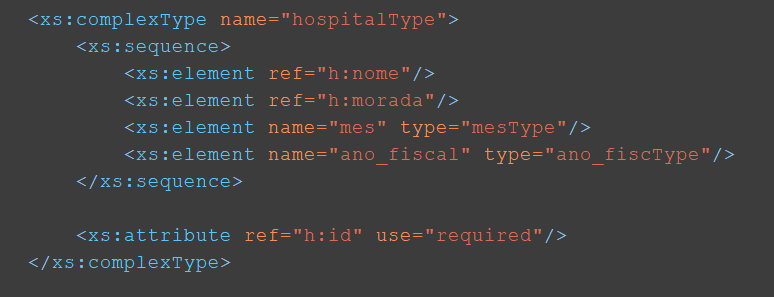
Por fim temos o elemento “lista\_registos”, que tem tem um tipo complexo, com uma sequencia de 1 elemento, com o elemento “registo\_clinico”, que é do tipo “rc:registoClinicoType” e esse elemento pode ocorrer no minimo 0 até no máximo infinitas vezes.



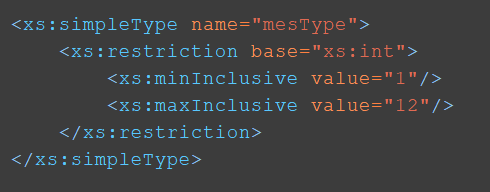
### InfoHospitalRegisto.xsd

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>"”. 

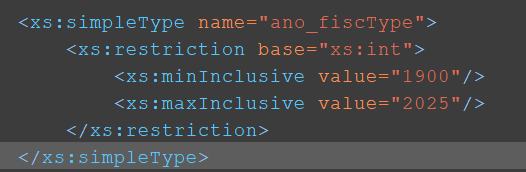
Depois definimos o complexType “hospitalType”, que possui uma sequencia com 4 elementos, sendo dois deles referência para o elementos do xsd “InfoGeralHospital.xsd”, sendo eles os elementos “h:nome” e “h:morada”, depois temos o elemento “mes”, que é do tipo “mesType” e o elemento “ano\_fiscal”, que é 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.



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.

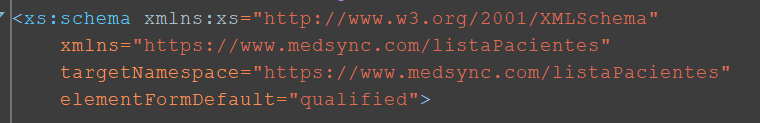


O “ano\_fiscType” é um sumpleType de tipo inteiro, que aceita apenas valores inteiros entre 1900 e 2025.

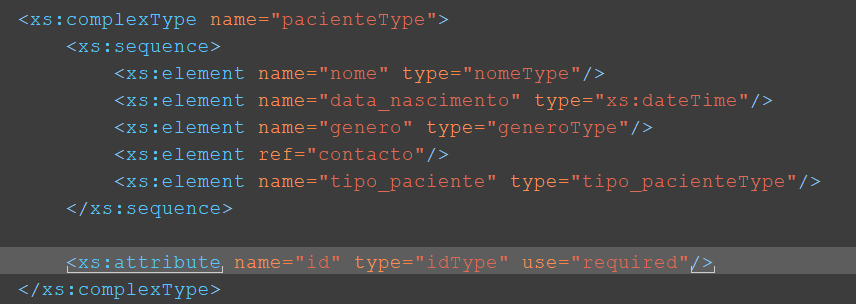


### PacientesAtendidos.xsd

Como dá para observar na imagem abaixo, começamos por definir um targetNamespace ao “PacientesAtendidos.xsd”.

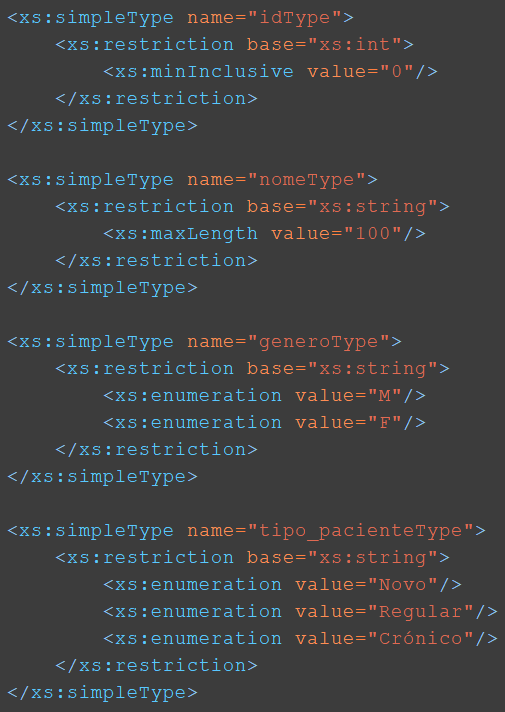


Após definir o targetNamespace, criamos o complexType “pacienteType”, que vai definir a estrutura de qualquer elemento do seu tipo. Sendo que o mesmo possui uma sequência com 5 elementos, começando com o elemento “nome”, que é do tipo “nomeType”, de seguida temos o elemento “data\_nascimento”, que vai 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 resaltar que o “pacienteType” ainda possui um atributo id que é do tipo “id\_type” e que é de uso obrigatório.



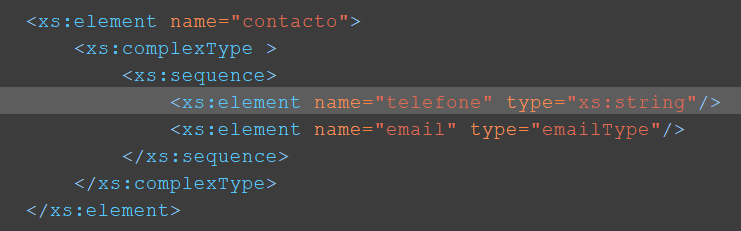
Na imagem abaixo é possível observar 4 simpleType, que foram utilizados no complexType “pacienteType”, sendo eles:

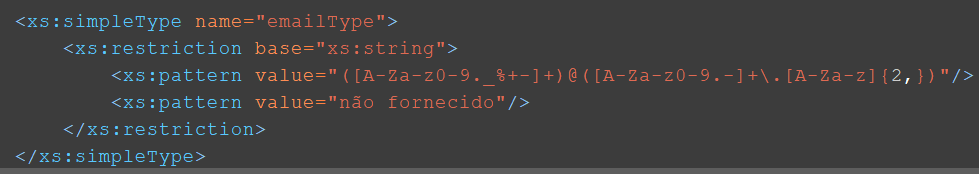
* “idType”, este simpleType é do tipo “xs:int” e ele só permite receber valores maiores ou iguais a 0;
* “nomeType”, este simpleType é do tipo “xs:string” e ele só pode receber no máximo 100 caracteres;
* “generoType”, ele é do tipo “xs:string” e ele possui duas enumerations, o que faz com que ele só possa receber dois valores o “M” ou “F”.
* “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”.



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”.

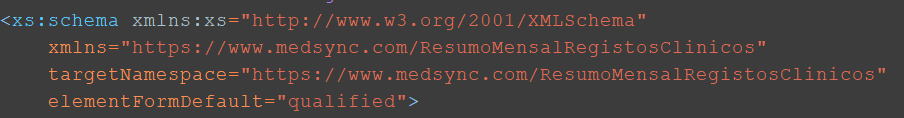


Por fim, para terminar este XSD, temos na imagem abaixo o simpleType emailType, que é restringido ao tipo xs:string e possui dois pattern, cada um indicando quais são os valores possíveis para o e-mail ser considerado válido. O primeiro pattern (<xs:pattern value="([A-Za-z0-9.\_%+-]+)@([A-Za-z0-9.-]+\.[A-Za-z]{2,})"/>) 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”.

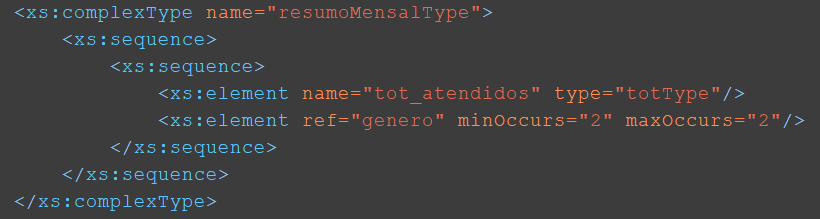
### RegistoClinico.xsd

### ResumoMensal.xsd

Mais uma vez como em todos os xsd começamos por especificar o targetNamespace do mesmo.

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.

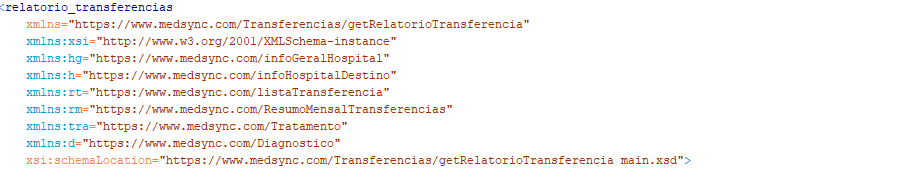
Na imagem abaixo dá para observar o simpleType “totType” ele é restringido para ser do tipo “xs:int”

## RelatorioTransferencias.XML

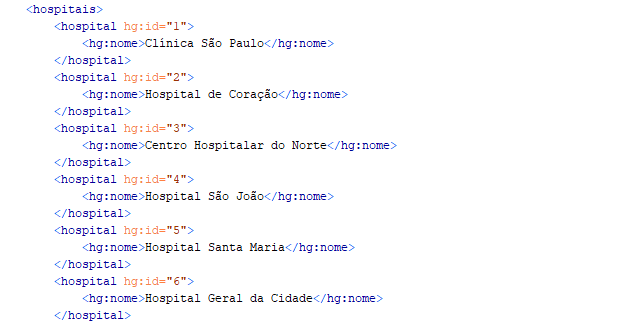
(Falar da estrutura e meter print)

A estrutura xml das Transferencias, deste exemplo está divida em três elementos, sendo eles:

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



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/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”.



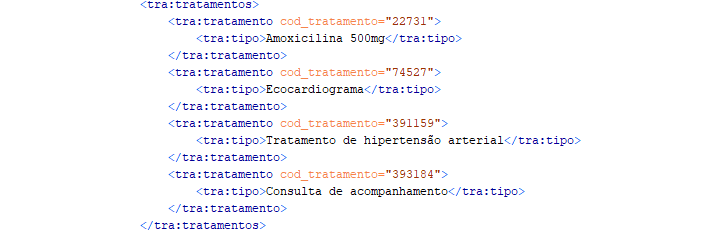
Acima dá para observar o elemento “hospitais”, que é o 1º elemento deste xml e armazena a informação de todos os hospitais (id que é um atributo e nome), de todos os hospitais que realizaram transferências no ano e mês em que se deseja fazer a pesquisa.

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 é armazenado 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;

.





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 seguinte 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;

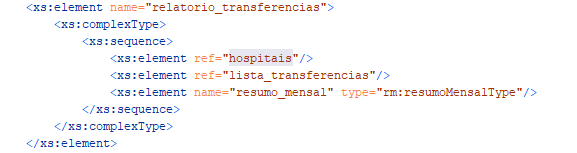


### main.xsd

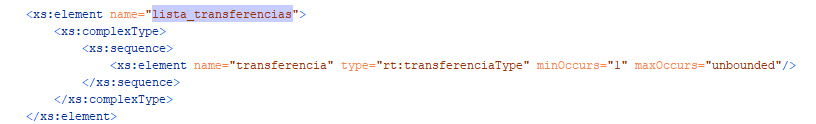
Como dá para observar na imagem abaixo, no main.xsd começamos por declarar os namespace e importar os mesmos.



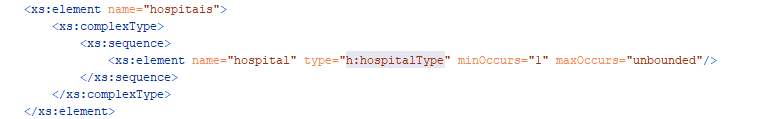
Depois criamos o elemento relatorio\_transferencias, como dá para observar na imagem abaixo, que é o elemento principal deste xsd. O mesmo possui um complexType e dentro do mesmo é definida a sequência dos seus elementos filhos. Onde vai haver um elemento “hospitais”, que está a ser referenciado, o elemento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



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-lo 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.

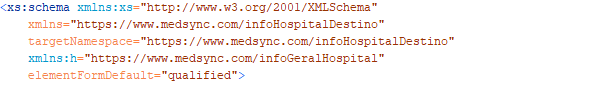


Por fim temos o elemento “hospitais”, que tem tem um tipo complexo, com uma sequência de 1 elemento, com o elemento “hospital”, que é do tipo “h:hospitalType” e esse elemento pode ocorrer no mínimo 1 e no máximo infinitas vezes.



### InfoHospital.xsd

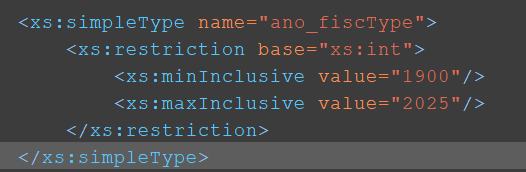
Como dá para ver na imagem abaixo, começamos por definir o targetNamespace e o xmlns deste xsd e inicializar o namespace “xmlns:h="https://www.medsync.com/infoGeralHospital"”.



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.



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



### Transferencias.xsd

(Falar da estrutura e meter print)

### ResumoMensal.xsd

(Falar da estrutura e meter print)

### ValoresGerais.xsd

(Falar da estrutura e meter print)

## XSD Gerais

(Meter texto do porquê criar XSD Gerais)

### InfoGeralHospital.xsd

### (Falar da estrutura e meter print)

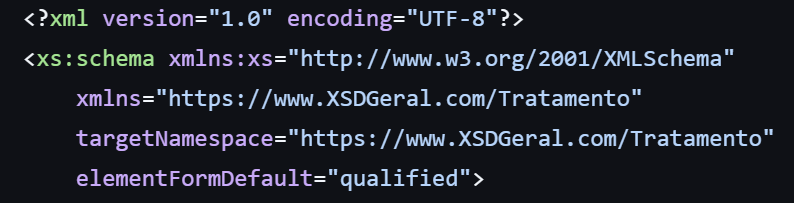
### Diagnosticos.xsd

(Falar da estrutura e meter print)

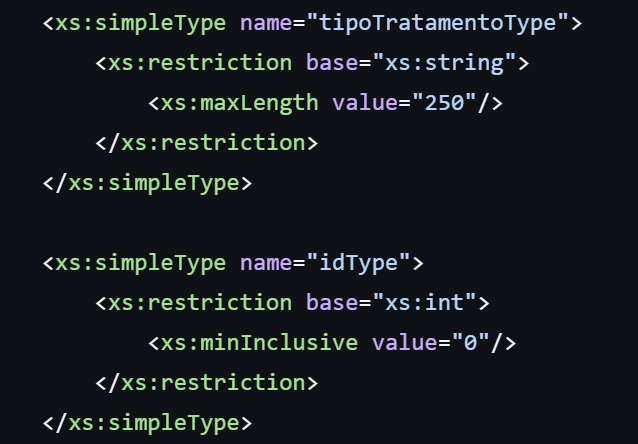
### Tratamentos.xsd

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*.



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.



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”.



# Mongo DB

(Meter texto a explicar o que é o MongoDB, dar introdução do que se vai falar no topicos e meter uma lista com os seus subtópicos)

## Modelação

(Meter texto a explicar as abordagens de modelação que usamos e indicar as collections que mudamos)

### Registos Clínicos

(Falar da modelação que fizemos nesta collection e meter print da query usada (Deixa eu meter Ass:Artur) que eu fiz duas separadas e o melhor é meter tudo numa e ainda vou fazer isso)

### Pacientes

(Falar da modelação que fizemos nesta collection e meter print da query usada)

## Functions

(Nos subTopicos falar do que faz as funções e meter prints)

### getPacientes

# 

### getRegistosClinicos

### getRelatorioRegistosClinicos

### getResumoMensalRegistosClinicos

### getResumoMensalRegistosClinicosPorGenero

### updateContactosPacientes

### updateTipoPaciente

### getHospitaisTransferencia

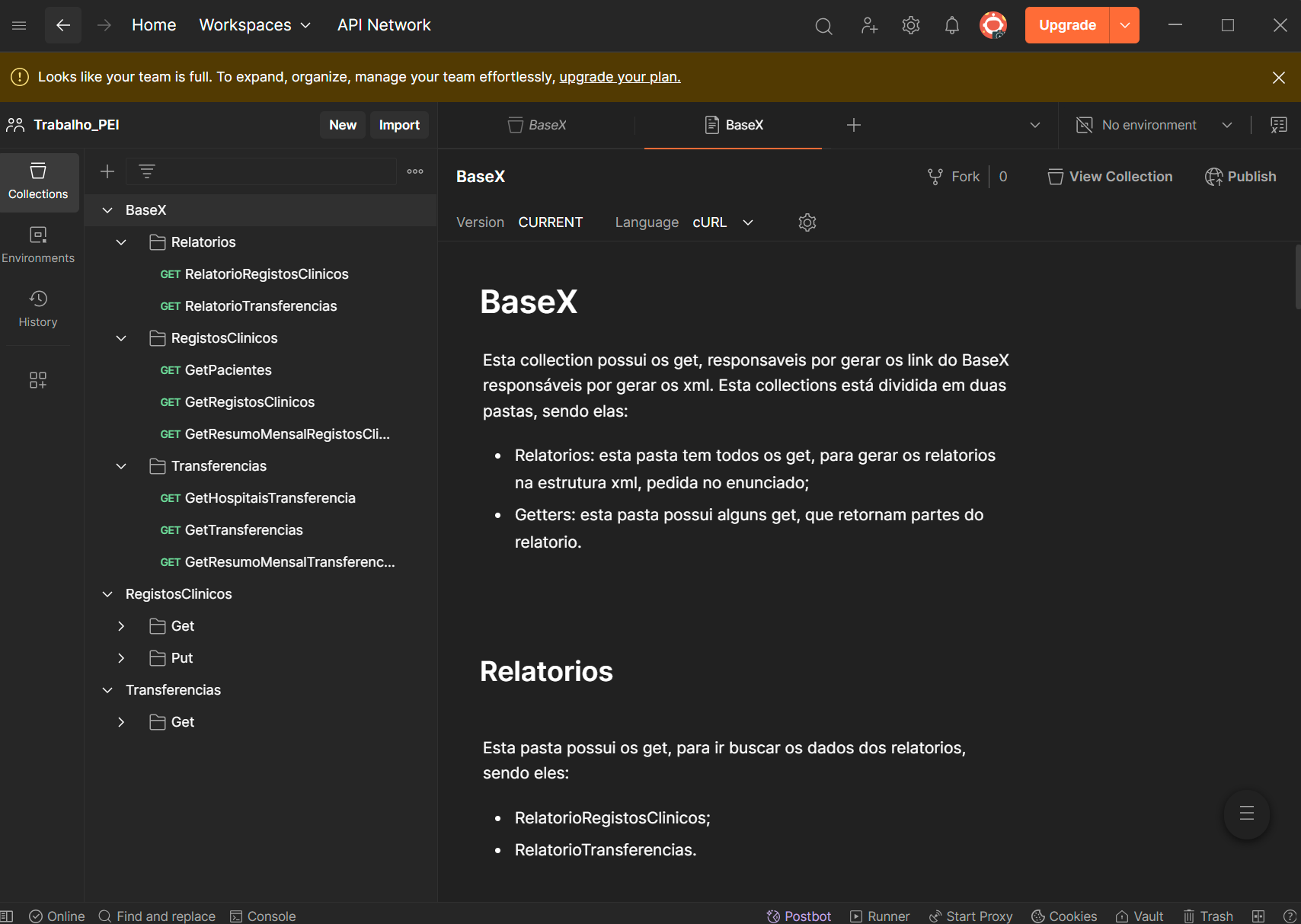
### getTransferencias

### getRelatorioTransferencias

### getResumoMensalTransferencia

# 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.



# BaseX

(Dar uma introdução ao topico)

## RelatorioRegistosClinicos.xq

(Meter print e falar um pouco)

## RelatorioTransferencias

(Meter print e falar um pouco)

# Conclusão

(Meter texto de conclusão)