Icon

Description automatically generated with medium confidence

# Capa

Unidade Curricular

Processamento Estruturado de Informação

Trabalho Prático

Docente: Bruno Moisés Oliveira

Alunos:

Rui Alexandre Borba Vitorino (8190479)

Bruno Miguel do Carmo Vieira (8190724)

Gilberto Jorge Da Mota Gomes (8210227)

Licenciatura em Segurança Informática de Redes de Computadores

# Índice

# Índice de Figuras

# Estrutura XML & XSD

A estrutura do trabalho prática assenta-se sobre 4 ficheiros XML estando a sua estrutura e composição definida em ficheiros XSD, que irão ser brevemente explicados.

* Atelier
* Reservation
* Family
* GeographicData
* CommonTypes

Começando a abordagem pelo último ficheiro, visto este ser utilizado por todo os outros; o ficheiro XSD CommonTypes contêm apenas dois tipos simples de elementos. Tipos estes que definem um padrão (pattern) para determinado tipo de dados, sendo essas padrões expressões regulares que capturam:

* Data (pe: 2022-01-15 e não 22/1/15)
* Nome (pe: “John Doe” e não “123John” ou “joão”)

Sendo o seguinte ficheiro utilizado pela Family.xsd e Reservation.xsd; o GeographicData contém alguns países e relativas informações para que se limite os dados introduzidos no momento da criação de uma reserva. Importante mencionar que não existe um elevado número de países devido a este ser apenas um exemplo, mas a seguinte referência contém o ficheiro base utilizados, este que sim, apresenta todos os dados. (*Core XML Schema - CountryType (Simple Type) | Ed-Fi Tech Docs*, n.d.)

Podendo uma reserva ter no máximo uma família, esta com um máximo de 7 elementos, viu-se vantajoso criar o ficheiro Family.xsd com as determinadas regras, podendo depois este ser referenciado aquando necessário.

Como mencionado, existe limitação de:

* Número máximo de famílias (uma família)
* Número máximo de elementos da família (sete elementos)
* Número de dias preferidos de reserva (cinco dias)

Como de esperado, além das regras, o ficheiro XSD define também a estrutura dos elementos da família, tal como nome, data de nascimento, o pais e cidade da família, existindo aqui referencia ao ficheiro GeographicData.

Sendo um dos dois principais ficheiros a serem utilizados posteriormente pela API e o *software* Postman, o Reservation.xsd contém a estrutura XML que é enviada como corpo de um pedido POST para a API, ou seja, todas as informações relativas a uma reserva a realizar.

Informações estas como a família, os seus elementos e respetivos dados, e os dias em que esta pretende visitar o atelier do Pai Natal.

Este ficheiro é de grande importância pois é aquele a ser verificado contra o esquema XSD, de forma a garantir o correto envio de informações e posterior armazenamento na base de dados.

Por último, o ficheiro Atelier.xsd é aquele que contêm as reservas em sí, com a indicação do dia e a respetiva disponibilidade (slots).

As restrições neste ficheiro são relativamente poucas, existindo apenas limite no formato da data (tal como existe em todos os outros ficheiros) e os valores permitidos nos *slots* (0 a 50, ambos inclusivos).

Inicialmente, este ficheiro contêm apenas a data inicial do funcionamento do Atelier do Pai Natal, mas à medida em que se realiza mais reservas, novas datas serão adicionadas, processo e decisão que irá ser explicada posteriormente no relatório.

# BaseX API

## Introdução

Servindo de base para toda a continuação do trabalho, uma API escrita em RestXQ e suportada pelo *software* BaseX foi criada tendo as seguintes funcionalidades:

### Funcionalidades

* Submissão de uma reserva
  + Após a verificação de disponibilidade em relação aos dias escolhidos e *slots* disponíveis
* Verificação de disponibilidade para um ou todos os dias
* Cancelamento de uma reserva (usando o seu ID).
* Exportação da Base de Dados

## Workflow

Para que todos os requisitos do enunciado fossem cumpridos, a API tornou-se complexa para quem a observa pela primeira vez, logo uma representação visual do *workflow* de execução e estruturação é essencial.

Diagram

Description automatically generated

Figura 1 Workflow de execução da API

## Em detalhe

Existindo agora um diagrama de facilita a visualização da execução da API, uma explicação detalhada de cada etapa e respetiva função pode ser efetuada.

1. post-xml

Parâmetro: $xml – Corpo do POST Request que contêm o XML da reserva

Como previamente mencionado, a primeira verificação a ser realizada é a validação do XML recebido contra o determinado esquema XSD Reservation, processo possível utilizando um modulo denominado “Validation”. (*Validation Module - BaseX Documentation*, n.d.)

Este processo garante que todos os dados estejam em conformidade, reduzindo consideravelmente os problemas em todas as seguintes etapas e a qualidade dos dados a serem armazenados.

No final da execução desta primeira função, a segunda função é chamada tendo-se passado a variável $xml e respetivo conteudo como argumento.

1. checks

Contendo a grande maioria das verificações, esta função irá primeiramente realizar as seguintes operações:

* Abrir a base de dados (com o comando db:open do modulo “Database”)
  + E armazenar o seu conteúdo em uma variável ($database)
* Armazenar dados em variáveis
  + As datas escolhidas da família e as datas presentes no atelier, respetivamente
  + $pfdates e $atelierdates

Embora os dados já estejam armazenados, é vantajoso filtrar-lhos de forma a melhorar as seguintes verificações.

A primeira filtragem, ou verificação, é encontrar todas as datas que a família escolheu **e** presentes no ficheiro Atelier; caso não se encontre nenhuma data em comum, a variável $existingDates estará vazia.

Para combater tal possibilidade, outra filtragem encontra todas as datas que a família escolheu **e não** estão presentes no ficheiro Atelier, pois relembrando o ficheiro Atelier apenas contêm a primeira data disponível, todas as outras terão de ser criadas e contêm disponibilidade (devido a não terem sido efetuadas reservas).

Assim, duas possibilidades já foram pensadas:

* A família escolheu dia(s) ainda não presente no Atelier
* A família escolheu dias(s) presentes no Atelier

Resta então filtrar uma última vez estes novos dados, capturando o primeiro dia em qualquer situação e realizar as duas seguintes verificações:

1º Verificação: O dia escolhido **e** presente no Atelier têm *slots* disponíveis?

*Pois, caso a data não exista, então haverá sempre disponibilidade*

2º Verificação: A data encontra-se no intervalo de tempo de 100 dias antes do Natal?

Efetuando então as verificações, começando por capturar todos os dias no Atelier em que existe disponibilidade (slots > 0) e comparar pelos dias que a família escolheu, caso exista, utiliza-se apenas o primeiro dia. Caso não existem dias em comum a variável $validDate estará vazia.

Aqui já se pode descartar que nenhum dia que a família escolheu está presente no Atelier (variável $newData) e/ou não existe disponibilidade para esses dias, sendo então apresentada uma mensagem de erro.

Este processo é efetuado verificando se ambas as variáveis estão vazias:

Text

Description automatically generated

Figura 2 Verificação se as variáveis estão vazias

Obviamente, só restam duas situações, existe uma data com disponibilidade no Atelier e que a família escolhera ou a família escolheu um dia que não existe no Atelier (e obviamente terá disponibilidade).

Parecendo uma boa situação para expor a decisão de se utilizar o ficheiro Atelier com apenas a data inicial e não ter todos os dias disponíveis no ficheiro XML; esta decisão é puramente por se achar que esse método ia contra uma correta e ciente solução que têm em conta a performance, sendo mais fácil verificar agora apenas uma data do que percorrer todo o documento procurando por uma similaridade, nesta situação seriam apenas 100 datas, mas se aplicarmos o mesmo pensamento para outras situações não estamos a aplicar boas práticas de desenvolvimento de uma API e *software*.

Assim sendo, resta verificar se a data, agora única, corresponde entre o intervalo de tempo entre os 100 dias antes do Natal (inclusive), passando a data como argumento para a função check-between-dates, retornando uma mensagem de erro caso esta não seja válida e parando todo o processo de criação de reserva.

Caso, a data seja válida **e** a data não exista no Atelier, esta é adicionada e a reserva é efetuada, processo realizado usando a função new-date, adicionando a data no final do ficheiro presente na base de dados e diminuindo um *slot* (pois corresponde a uma nova reserva).

A criação da reserva é simplesmente a adição de um ficheiro XML (este que é retornado pela função return-xml-reservation) com as informações da família, a data da reserva e o seu estado, bem como um ID, na base de dados com o auxilio do comando db:add.

Finalmente, de forma a dar *feedback* ao utilizador, uma mensagem é retornada, indicando o sucesso da criação da reserva e o respetivo ID; ID este que poderá ser usado para cancelar a reserva.

## Cancelamento

Através do ID retornado após a criação de uma reserva, é também possível realizar o seu cancelamento.

Acedendo ao URL /cancelreservation , indicando o ID como parâmetro e enviando um POST Request.

Exemplo: / cancelreservation?id=5 (irá cancelar a reserva com o ID 5)

Após o cancelamento da reserva, um slot é novamente adicionado ao dia em que a reserva estava associada, pois existe agora disponibilidade para mais uma família visitar o Atelier do Pai Natal.

### Particularidades

Foi adicionado um Try Catch que, caso a reserva não seja encontrada, isto pois o utilizador poderá indicar um ID que não esteja associado a nenhuma reserva, uma mensagem de erro é apresentada.

E se a reserva já tiver sido cancelada? É também uma situação esperada, e verificada, comparando o valor do campo do estado da reserva, caso tenha o valor “Canceled” então uma mensagem de erro é apresentada e a reserva não é novamente cancelada.

## Verificação de Disponibilidade

Enviando um GET Request ao seguinte URL, /availability, juntamente com uma data ou um valor específico, é possível retornar a disponibilidade para a(s) data(s) indicada(s).

Ou seja, caso se envie um GET Request com os seguintes parâmetros, ir-se-á apresentar a disponibilidade para os seguintes dias: 15, 16 e 17 de Dezembro.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 3 Verificação de disponibilidade para três dias específicos

Ao se passar o parâmetro “all” este irá mostrar a disponibilidade dos dias presentes no Atelier, caso o dia pretendido não esteja presente no resultado então é porque este ou não se encontra dentro do intervalo de tempo disponível ou está disponível e ainda não existem reservas no Atelier para tal dia.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 4 Verificação global de disponibilidade

# Python Scripts

Foram desenvolvidos dois *scripts* na linguagem de programação Python devido a dois elementos do grupo terem conhecimento sobre a linguagem e as suas capacidades pudessem ser enquadradas e utilizadas durante a realização do trabalho, nomeadamente nas tarefas:

* Geração aleatória de dados
* Transformação de ficheiros XML para JSON
* Importação dos ficheiros JSON para MongoDB
* Embutimento de informações em ficheiros JSON

## generateData.py

Este *script* tem como funcionalidade gerar dados aleatórios para que se possa tornar as reservas a analisar relativamente mais próximas a uma realidade.

Dados estes que são compreendidos entre uma quantidade aleatória de elementos da família (entre 1 a 7), sendo os seus nomes e datas de nascimento também aleatórias, e um número de datas preferidas de visita ao Atelier (entre 1 a 5), compreendidas dentro do intervalo de tempo de funcionamento do Atelier.

O *script* é executado através da linha de comandos, tendo dois parâmetros:

1. Número de reservas a gerar
2. Número de reservas a cancelar

Após o envio dos dados para a API e o seu armazenamento, é ainda possível executar o processo de exportação da base de dados, este que envia um GET Request à API e esta indica ao BaseX o que realizar.

## convert.py

Este *script* tem dois propósitos, estando estes interligados.

O primeiro é a remoção de *namespaces* e prefixos dos ficheiros XMLs exportados pelo BaseX (visto este apenas conseguir exportar os ficheiros armazenados na sua base de dados de forma correta em XML), utilizando expressões regulares, gerando um XML simples e puro para posterior conversão para JSON.

Conversão esta que é efetuada usando o modulo Python xmltodict (*Xmltodict · PyPI*, n.d.).

# MongoDB

Text

Description automatically generated