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.xsd 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.xsd 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.)

Face às regras impostas, e às especificidades atribuídas aos elementos que compõem uma família, optou-se por criar um XSD para validação dos dados introduzidos, denominado por “Family.xsd ”, podendo depois este ser referenciado aquando necessário.

Como mencionado, existe limitação de:

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

Define a estrutura dos elementos:

* Nome
* Data de nascimento
* País
* Cidade (ficheiro GeographicData)
* Contacto de emergência

Todas as informações referentes a cada reserva são colocadas numa estrutura XML, para ser enviada como “corpo” de um pedido POST para a API, as mesmas são validadas pelo Reservation.xsd.

Informações contidas em cada reserva:

* Todos os dados da família
* Datas de preferência

Este ficheiro é de grande importância pois a correta validação do mesmo vai garantir a conformidade dos dados que posteriormente serão adicionados à Base de Dados.

Por último, o ficheiro Atelier.xsd é aquele que contêm as datas disponíveis para visita e que faz a gestão do limite diário de 50 famílias.

Informações contidas em data:

* Data (validada dentro dos 100 dias)
* Número de visitas disponíveis (slots)

As restrições implementadas neste ficheiro são o formato da data, que é do tipo xs:date, com um padrão definido por uma expressão regular, de forma a garantir a conformidade das datas, pois à posterior, estes dados serão usados numa função de validação de intervalo temporal (100 dias antes do Natal, “2022-09-17 a 2022-12-25”).

Os valores permitidos nas *slots* (0 a 50, ambos inclusivos), é uma validação redundante, uma vez que sempre que é criada ou cancelada uma reserva, o valor da *slot* é alterado por um processo de validação automático.

Inicialmente, este ficheiro contém apenas a data inicial do funcionamento do Atelier do Pai Natal, dia 2022-09-17 com 50 *slots* disponíveis, no entanto, sempre que se realizam novas reservas, as datas serão adicionadas num processo automático e decisão, que será analisado posteriormente na explicação da API.

# BaseX API

## Introdução

Servindo de base para toda a continuação do trabalho, a 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.
* 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

Com a ajuda do diagrama torna-se mais simples a explicação do funcionamento da API, que será apresentado de forma detalhada nos pontos seguintes:

1. post-xml

Parâmetro: $xml – Contém o “Body”do POST Request que corresponde ao XML da reserva

1. Validação do XML com o Reservation.xsd utilizando a função “validate:xsd()”

(*Validation Module - BaseX Documentation*, n.d.), se os dados não cumprirem com a estrutura definida, dá um erro, se estiverem corretamente formatados, então continua para o ponto 2.

1. É chamada a função checks(), com a variável $xml passada como argumento com o XML já validado.
2. checks

É a função mais importante da API, pois será a responsável por:

* filtrar e validar as datas
* Adicionar reservas à base de dados
* Fazer updates à base de dados

1. Variáveis declaradas na função check()

|  |  |
| --- | --- |
| NOME VARIÁVEL | CONTEÚDO/OBJETIVO |
| $database | Aceder à informação contida na Base de Dados |
| $pfDates | Datas preferidas da família |
| $atelierDates | Datas presentes no ficheiro Atelier.xml |
| $existingDates | Datas escolhidas pela família e que já existem no ficheiro Atelier.xml |
| $validSlots | Datas contidas no $existingDates que têm “slots” com o valor maior do que zero. (filtra os dias escolhidos pela família que têm disponibilidade) |
| $validDate | Primeira data da $validSlots, caso não esteja NULL e passe em todas as validações seguintes, será utilizada para criar uma reserva. |
| $notExistingDates | Datas escolhidas que não constam no Atelier.xml.  Caso esta seja uma data válida, entre 2022-09-15 e 2022-12-23 será usada para a criação automática da data. |
| $checkBetweenDates | Recebe o resultado da validação da $notExistingDates |
| $newDate | Primeira data da $checkBetweenDates caso não esteja NULL, será utilizada para adicionar uma nova data ao ficheiro Atelier.xml e criar uma reserva. |
| $rid | Contém o valor do total de reservas mais um. |

1. Explicação do processo de filtragem da informação

Uma vez que foi tomada a decisão de as datas para os dias das visitas serem criadas num processo automático, será necessário efetuar uma série de verificações, que garantam o sucesso desta opção.

Uma vez que a família pode escolher datas que ainda não existam no ficheiro Atelier.xml, é necessário validar se essas se encontram nos 100 dias definidos para as visitas.

A Base de Dados é inicializada apenas com o primeiro dia do intervalo de tempo definido, ou seja o ficheiro Atelier.xml contém apenas uma data disponível.

Assim, existem várias possibilidades:

* A família escolher dia(s) ainda não presentes no Atelier.xml mas válidos no intervalo
* A família escolher dia(s) ainda não presentes no Atelier.xml e in válidos no intervalo
* A família escolher dias(s) presentes no Atelier.xml com disponibilidade
* A família escolher dias(s) presentes no Atelier.xml sem disponibilidade

Inicia-se então um processo de filtragem que permita validar ou não as datas preferidas da família.

1ª Verificação: Encontrar todas as datas que a família escolheu **e** que estão presentes no ficheiro Atelier.xml, caso não existam datas em comum, a variável $existingDates, estará vazia.

2ª Verificação: Verifica se as datas da família ainda não existem no Atelier ($notExistingDates).

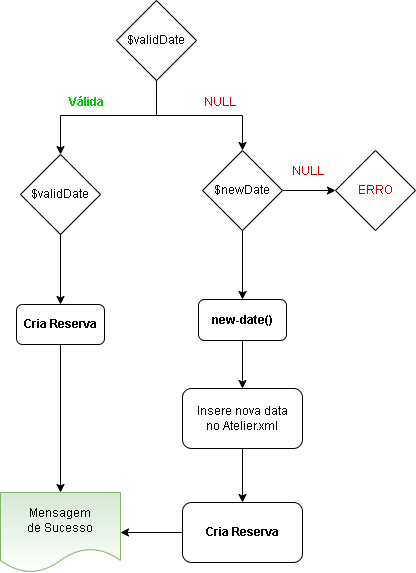
3ª Verificação: Validar se as datas da $notExistingDates estão dentro do intervalo.

1. Processo de tomada de decisão

Se a variável $validDate estiver vazia e a $newDate também é porque a Oficina do Pai Natal atingiu o máximo de capacidade para o(s) dia(s) escolhidos ou as datas escolhidas não estão entre 2022-09-15 e 2022-12-23

Se a variável $validDate estiver vazia e a $newDate não, então é porque a data não existe no Atelier.xml, logo a mesma será adicionada e será criada uma reserva.

Se a variável $validDate não estiver vazia, é porque existia no Atelier.xml uma data coincidente com uma das datas preferidas da família e com slots disponíveis, então é criada uma reserva.



1. check-between-dates(arg1)

Faz-se uma iteração por cada data recebida no argumento e verifica se esta se encontra entre 2022-09-15 e 2022-12-23, retornando as válidas ou retornando a variável vazia se estiverem fora do intervalo.

1. new-date(arg1)

Abre o Atelier.xml e insere um novo elemento "reservations" com a data que é passada no argumento já com as slots a 49, uma vez que vai ser criada a primeira reserva para este dia.

1. return-xml-reservation(arg1,arg2)

Com a informação passada no primeiro argumento elabora e retorna um XML com a estrutura de uma reserva para a data que está no segundo argumento.

Para atribuir o id a esta nova reserva conta as já existentes e incrementa um valor.

## 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