Relatório de PEI

Estrutura do projeto

- `/data` : Base de dados
- `/migration` : Script de migração
- `/mongo` : Query's
- `/postman` : Coleção de endpoints
- `/restxq` : Rest API
- `/xml` : Exemplos de XML
- `/xsd` : XSD Schemas

Links

Mongo Charts

Contextualização e Caracterização do caso de estudo

No âmbito da unidade curricular `Processamento Estruturado de Informação` para efeitos de avaliação contínua realizámos um trabalho prático com a finalidade de fornecer uma REST API ao "Pai Natal" de forma a facilitar a informação necessária para a realização de agendamentos das famílias.

Sabemos ainda que o "Pai Natal" decidiu disponibilizar visitas à sua oficina 100 dias antes do natal porém, a demanda é imensa fazendo com que o "Pai Natal" desejasse tornar o processo de agendamento o mais justo e eficaz possível, para satisfazer esta necessidade foi elaborada uma restrição com um número máximo de 5.000 famílias sendo apenas possível a visita de 50 famílias por dia.

Assim, nós o grupo nº1, os entusiastas da programação, decidiram ajudar o "Pai Natal" criando uma `API` com recurso a tecnologias como: `BaseX`, `MongoDB`, `Postman` e linguagens: `XQuery`, `XPath`, `Python`.

No contexto real temos o `Twitter` e o `Instagram` onde os dois "gigantes da tecnologia" utilizam o mesmo método de pesquisa (procura pelo `ID`) que o projeto aqui desenvolvido, como por exemplo na função `Cancel` onde o `ID` é passado como parâmetro para depois ser procurado dentro da base de dados.

Abordagem do problema

Conforme o enunciado nos indica para efetuar uma reserva é necessário explicitar as datas de preferência da família e os respetivos elemetos que constituem o agregado familiar até 7 membros, ou seja cada elemento da família deve introduzir o seu `Name`, `Country`, `City` e `Birthday`.

De forma a agilizar o processo de agendamento das visitas à oficina do "Pai Natal", desenvolve-mos uma REST API em BaseX com vários endpoints que permitem de forma fácil adicionar e remover reservas bem como verificar a disponibilidade entre duas datas.

Esta REST API consome um ficheiro XML que posteriormente é validado contra um ficheiro XSD Schema e mais tarde estes dados são armazenados na base de dados do `BaseX` e no final é devolvido o id da reserva autogerado e único.

Para complementar desenvolve-mos também um dashboard analítico com recurso ao serviço `Mongo Charts` de forma a promover uma melhor visualização dos dados das reservas, estas ínclui:

- `Average of persons per day`
- `Number of cancellations per day`
- `Percentage of occupation per day`
- `Total bookings country`
- `Total bookings per city`
- `Sum of bookings until today`
- `Number of persons by age group`
- `Total of Families per day`

Foi também desenvolvido um script de migração em linguagem `Python` que agiliza o processo de migração entre a API do `BaseX` e o `MongoDB`.

Configurações

Inicialização da API em BaseX

Deve executar os seguintes comandos na linha de comandos:

```
$ cd PEI_TP_AC
$ chmod +x ./start.sh
$ ./start.sh
````

Execução do script de migração
No seguinte excerto deve ser criado um ficheiro com o nome `.env`
com a key `MONGODB_URI` e a `connection string` como value.

.env

MONGODB_URI=

```
Deve executar os seguintes comandos na linha de comandos:
```shell
$ cd PEI_TP_AC/migration
$ pip3 install -r requirements.txt
$ python3 migration.py
```

Identificação das propriedades do XSD

Namespaces

No seguinte excerto de código declaramos o `targetNamespace` cuja declaração tem como significado que todos os elementos (filhos do root) do documento pertencem ao mesmo namespace, é normal a utilização do atributo `elementFormDefault` na definição do XSD com a finalidade de indicar que todos os elementos são `qualified` ou seja que estão associados ao target namespace.

No excerto de código a seguir descreve o namespace por defeito para indicar que todos os elementos utilizados neste documento estão declarados no namespace.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="https://www.w3schools.com/ReservationSchema"
           elementFormDefault="qualified"/>
```

```
 <?xml version="1.0" encoding="UTF-8" ?>
<Reservation xmlns="https://www.w3schools.com/ReservationSchema">
</Reservation>
```

Elementos do XSD

Este excerto de código permite a definição do elemento `Reservation` com os seus respetivos elementos `Family` e `Days` e os seus respetivos tipos.

```
<xs:element name="Reservation">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Family" type="rs:FamilyType"/>
            <xs:element name="Days" type="rs:DaysType"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<Reservation>
    <Family>
        <!--1 to 7 repetitions:-->
        <Member>
            <Name>string</Name>
            <Country>string</Country>
            <City>string</City>
            <Birthday>2008-09-29</Birthday>
        </Member>
    </Family>
    <Days>
        <!--1 to 5 repetitions:-->
        <Day>2021-12-23</Day>
    </Days>
</Reservation>
```
```

Tipos Complexos

FamilyType

O seguinte excerto de código permite a definição de uma sequência com o valor máximo de 7 membros.

```
<xs:complexType name="FamilyType">
    <xs:sequence>
        <xs:element name="Member" type="rs:MemberType" maxOccurs="7"/>
    </xs:sequence>
</xs:complexType>
```

MemberType

O seguinte excerto de código permite definir o tipo complexo `MemberType` que constituí uma sequência de strings tais como: `Name`, `Country`, `City`, `Birthday`.

```
<xs:complexType name="MemberType">
    <xs:sequence>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Country" type="xs:string"/>
        <xs:element name="City" type="xs:string"/>
        <xs:element name="Birthday" type="xs:date"/>
    </xs:sequence>
</xs:complexType>
```

DaysType

O seguinte excerto de codigo permite definir uma sequência máxima de datas até 5 dias.

```
<xs:complexType name="DaysType">
    <xs:sequence>
        <xs:element name="Day" type="rs:DayType" maxOccurs="5"/>
    </xs:sequence>
</xs:complexType>
```

Tipos Simples

DayType

O seguinte excerto de codigo define um tipo simples chamado `DayType` com uma restrição. Este deve possuir exatamente 10 caracteres, a data deve estar compreendida entre `2021-10-16` e `2021-12-25`.

```
<xs:simpleType name="DayType">
    <xs:restriction base="xs:date">
        <xs:minInclusive value="2021-10-16"/>
        <xs:maxInclusive value="2021-12-25"/>
        <xs:pattern value=".{10}"/>
    </xs:restriction>
</xs:simpleType>
```

Coleção do MongoDb

No seguinte documento `JSON`, é possível visualizar toda a informação referente a uma reserva:

- `/Id` : Código da reserva. `Tipo de dados` : Inteiro

- `/Canceled` : Representa o estado da reserva, "true" significa que a reserva encontra-se cancelada. `Tipo de dados` : Boolean

- `/NumberOfMemebers` : Número total de membros existentes na reserva. `Tipo de dados` : Inteiro

- `/ScheduleDate` : Demosntra a data e hora da reserva. `Tipo de dados` : Date

- `/Name` : Nome do membro. `Tipo de dados` : String

- `/Birthday` : Representa a data de nascimento do membro. `Tipo de dados` : Date

- `/City` : Representa a cidade do membro. `Tipo de dados` : String

- `/Country` : Representa o país do membro. `Tipo de dados` : String


```
{
  "_id": {
    "$oid": "61e5935c0f6e891675adbe05"
  },
  "Id": 43,
  "Canceled": true,
  "NumberOfMembers": 4,
  "ScheduleDate": {
    "$date": "2021-12-20T00:00:00Z"
  },
  "City": "Porto",
  "Country": "Portugal",
  "Members": [
    {
      "Name": "Josefina Fonseca",
      "Country": "Portugal",
      "City": "Porto",
      "Birthday": {
        "$date": "1980-07-11T00:00:00Z"
      }
    },
    {
      "Name": "Aur\u00e9lio Ferreira",
      "Country": "Portugal",
      "City": "Porto",
      "Birthday": {
        "$date": "1972-03-26T00:00:00Z"
      }
    },
    {
      "Name": "Gertrudes Ferreira",
      "Country": "Portugal",
      "City": "Felgueiras",
      "Birthday": {
        "$date": "2017-11-10T00:00:00Z"
      }
    },
    {
      "Name": "Gertrudes Ferreira",
      "Country": "Portugal",
      "City": "Felgueiras",
      "Birthday": {
        "$date": "2015-07-31T00:00:00Z"
      }
    }
  ]
}
```


REST API

Adionar Reserva

Na função `Add` é utilizado a função `Availability` onde a mesma será utilizada para a verificação da disponbilidade entre duas datas onde após isso será adicionada uma reserva na data pedida pelo utilizador caso a mesma esteja disponível.

Disponibilidade das Reservas

A função `Availability` verifica a disponibilidade entre duas datas para que seja possível obter a informação das datas disponíveis para efetuar uma reserva.

Cancelar uma Reserva

Na função `Cancel` o `Id` da reserva é passado como parâmetro, de seguida o mesmo é procurado dentro da base de dados onde caso seja econtrado o estado de `Canceled` é passado para `true`, na hipótese de uma reserva se encontrar cancelada e existir a tentativa de cancelar novamente a mesma se manterá com o estado `true` não havendo quaisquer mudanças.

Inicializar Base de Dados

A função `Init` é utilizada para a inicialização da base de dados onde também é possível fazer um reset à mesma.

Listar Reservas

Na função `List` é possível visualizar todas as reservas efetuadas que se encontram na base de dados.

Análise Crítica

Como último ponto deste relatório, mas não menos importante, a análise crítica é um elemento fundamental, a medida em que permite uma reflexão sobre a forma como este projeto foi desenvolvido, sobre o cumprimento ou não dos objectivos:

- [x] Desenvolver um vocabulário em XSD.
- [x] Desenvolver uma REST API em BaseX.
- [x] Desenvolver um Script de migração.
- [x] Efetuar Query's no Mongo Atlas.
- [x] Desenvolver um dashboard no Mongo Charts.

De uma forma global, julgamos serem estes os objetivos principais desta Unidade Curricular.

Fazendo agora um balanço do que foi realizado ao longo deste projeto, podemos afirmar que uma grande parte dos objetivos foram atingidos de uma forma bastante satisfatória.

Por outro lado, verificam-se algumas dificuldades no desenvolvimento da API em BaseX devido à falta de documentação existente numa tecnologia consideravelmente antiquada, embora que os docentes tenham se mostrado sempre disponiveis para responder as eventuais duvidas que vão surgindo.

No que diz respeito ao MongoDB, foi bastante enriquecedor a aprendizagem de uma nova tecnologia de base de dados não relacional que no futuro irá se revelar uma aptidão que nos permitirá responder da forma mais adequada aos problemas nos iremos deparar nos tempos que se aproximam.

Terminamos o relatório deixando uma frase, `Toda a gente tem uma API até o Pai Natal!`