P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

2022

PEID – Trabalho Prático

PROCESSO, ESTRUTURAÇÃO E INTEGRAÇÃO DE DADOS

ANDRÉ PINTO (8200613) / SERGIO FÉLIX (8200615)

Índice

Introdução 3
Implementação 4
XML/XSD 4
BaseX / RestQX 4
Migração 5
Consultas 6
MongoDB Charts 9
Conclusão 11
Bibliografia 12

Figura 1 - Parte do código do schema....4
Figura 2 - Função de conversão de XML para JSON....5
Figura 3 - Consulta que visa apresentar o número total de visitantes com idade superior a 50 anos....6
Figura 4 - Consulta que visa apresentar por localidade o número de brindes oferecidos de cada tipo....6
Figura 5 - Consulta que visa apresentar a sala que em média os visistantes demoram mais tempo....7
Figura 6 - Consulta que visa apresentar os 5 produtos mais vendidos....7
Figura 7 – Consulta que visa apresentar a média de amigos levados por visita....7
Figura 8 - Consulta que visa apresentar a distribuição dos visistantes por localidade....8
Figura 10 - Número de visitantes com idade maior de 50....9
Figura 9 - Média Amigos por Visita....9
Figura 11 - Top 5 Produtos mais vendidos....9
Figura 12 - Média de Tempo despendido por cada sala....9
Figura 13 - Gráfico Heat Map Distribuição....10
Figura 14 - Grafico GRID Número de brindes oferecidos por localidade e tipo....10

Introdução

No âmbito da disciplina de Processo, Estruturação e Integração de Dados do 2º ano, 1º semestre do Curso Técnico Superior Profissional de Desenvolvimento para a Web e Dispositivos Moveis, foi pedido o desenvolvimento de um sistema que permita rastrear o tempo despendido por cada visitante em cada sala do museu, contudo apenas se aplicando aos visitantes que estivessem dispostos a responder um questionário em troca de um pequeno brinde alusivo ao museu como forma de agradecimento.

Estre trabalho, será iniciado com a criação da estrutura do questionário em XML Schema Definition (XSD) que posteriormente será usado para validar a inserção dos dados na REST API desenvolvida em XQuery. Na API será necessário o desenvolvimento de algumas rotas cruciais bem como alguma lógica extra para as mesmas.

Posteriormente será necessário migrar os dados para uma base de dados orientada por documentos utilizando MongoDB de forma a satisfazer as necessidades da estruturação e organização de dados, conforme os requisitos apresentados anteriormente.

Por fim desenvolver algumas consultas relevantes que irão posteriormente ser convertidas em gráficos no MongoDB Charts possibilitando assim uma fácil análise dos dados.

Implementação

XML/XSD

De modo a responder a todos os pontos pedidos foi criado uma estrutura, estrutura esta que nos vai possibilitar futuramente na API validar a introdução dos dados. A mesma possui todos os elementos essenciais, porém, alguns deles com restrições para delimitar os valores que podem ser introduzidos.

No elemento de preço contido no elemento produto, existe uma restrição que apenas é permitido preços até 8, já no elemento brinde uma restrição existente é a de apenas permitir a escolha de 4 preferências de brindes.

```
<xs:element name="compras">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="produto" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="descricao" type="xs:string" />
                        <xs:element name="preco">
                            <xs:simpleType>
                                <xs:restriction base="xs:double">
                                    <xs:maxInclusive value="8" />
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="brindes">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="brinde" type="xs:string" minOccurs="0" maxOccurs="4" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Figura 1 - Parte do código do schema

BaseX / RestQX

Como pedido no enunciado, foi necessário desenvolver uma API em RestXQ que contemplasse todos os pontos pedidos. Para tal foi necessário a criação de 3 rotas distintas:

• "visita/add" – Esta é a rota principal. Nela é possível submeter os dados do questionário da visita. Esta rotas recebe por POST os dados necessários. Após validar os dados introduzidos através do XSD, gera um id único para identificar a visita. Este ID é obtido através do maior ID existente mais um. De seguida passa para o cálculo do brinde oferecido. Por ordem de preferência do utilizador, caso o item já tinha sido comprado, é atribuído caso contrário é atribuído um brinde default que é um porta-

chaves. Por fim os dados são inseridos na base de dados e uma resposta de sucesso é retornada au utilizador.

• "visita/produtos/{$who}" – Esta rota retorna a lista dos produtos comprados. Foi uma rota inicialmente criada para facilitar na criação da rota anterior.

• "visita/exports" – Esta rota também é muito importante. Nela obtemos todos os dados presentes na base de dados embebidos numa tag global. Esta rota permite que a script em Python obtenha os dados necessário para realizar a migração.

Migração

No momento de migração de dados para o MongoDB, foram desenvolvidos dois métodos em consequência de um problema encontrado.

No primeiro método, os dados foram moldados em volta de scripts desenvolvidos no Mongo Shell, os dados foram transpostos em um website conversor XML para JSON online, após a execução dos scripts ao analisar o resultado um problema foi encontrado, sendo este que quando um array continha vários objetos o mesmo criava um array de objetos, porém, se apenas existisse um array com um objeto o mesmo moldava-o como um objeto de um objeto não correspondendo as nossas espectativas.

```
def xml_to_json(xml_document):
    output = []
    visits = xml_document.visitas.findAll('visita')
    for visit in visits:
        _visit = {
            "@id": int(visit['id']),
            "date": convert_string_to_datetime(visit.data.string),
            "numAmigos": int(visit.numAmigos.string),
            "localidade": visit.localidade.string,
            "idade": int(visit.idade.string),
            "compras": [],
            "brindes": [],
            "gift": visit.gift.string,
        }
        purchases = visit.compras.findAll('produto')
        for purchase in purchases:
            _purchase = {
                "descricao": purchase.descricao.string,
                "preco": float(purchase.preco.string),
            }
            _visit["compras"].append(_purchase)
        gifts = visit.brindes.findAll('brinde')
        for gift in gifts:
            _gift = gift.string

            _visit["brindes"].append(_gift)

        output.append(_visit)

    return output
```

Figura 2 - Função de conversão de XML para JSON

De forma a contornar o problema, o grupo decidiu implementar um algoritmo em Python que executava uma request a uma das rotas na API que exportava os dados, após isso, convertia os mesmos em formato JSON, porém, desta vez corretamente, como tal se verificava os mesmos eram também logo inseridos na base de dados.

Por fim é executada a script para associar o distrito á coleção localidades. Deste modo o distrito passa do tipo String para tipo ObjectId que referência a respetiva localidade que está armazenada na coleção localidades.

Assim fica concluída a migração dos dados.

Consultas

Na primeira consulta requisitada ao grupo não foram encontradas nenhumas dificuldadesonsulta baseia-se em mostrar o número de visitantes com idade superior a 50 anos.

```
db.visitas.aggregate([
    {
        $match: {
            idade: {
                $gt: 50,
            },
        },
    },
    {
        $count: 'Número total de visitantes com mais de 50:',
    },
]);
```

Figura 3 - Consulta que visa apresentar o número total de visitantes com idade superior a 50 anos

Como tal foi apenas necessário um simples *count* que fosse de acordo com o *match* que verifica as idades que são maiores que (greater than - $gt) 50.

A segunda consulta referente a apresentação do número de brindes oferecidos de cada tipo por localidade, em decisão unânime, foi considerada a mais complicada, pois foi a consulta que foi necessário dividir por etapas, e fazer algumas pesquisas que alocou mais tempo dos dois integrantes até conseguir obter o resultado correto.

Inicialmente foi agrupado por localidade e tipo de brinde, posteriormente agrupando por localidade contendo o tipo de brindes bem como a sua contagem.

Como a estrutura dos documentos esta dividida entre visitas e localidades foi necessário realizar um *lookup* para mostrar na projeção o distrito ao invés do *ObjectId*.

```
db.visitas.aggregate([
    {
        $group: {
            _id: {
                localidade: '$localidade',
                gift: '$gift',
            },
            sum: {
                $sum: 1,
            },
        },
    },
    {
        $group: {
            _id: '$_id.localidade',
            gifts: {
                $push: {
                    name: '$_id.gift',
                    count: '$sum',
                },
            },
        },
    },
    {
        $lookup: {
            from: 'locations',
            localField: '_id',
            foreignField: '_id',
            as: 'localidade',
        },
    },
    {
        $project: {
            cidade: {
                $arrayElemAt: ['$localidade.city', 0],
            },
            gifts: '$gifts',
        },
    },
]);
```

Figura 4 - Consulta que visa apresentar por localidade o número de brindes oferecidos de cada tipo

Por fim, sem grandes dificuldades na última consulta proposta referente a apresentação da sala que em média os visitantes demoram mais tempo o grupo conseguiu obter os resultados esperados.

```
db.visitas.aggregate([
    {
        $unwind: '$tempos',
    },
    {
        $group: {
            _id: '$tempos.sala',
            average: {
                $avg: '$tempos.tempo',
            },
        },
    },
    {
        $sort: {
            average: -1,
        },
    },
    {
        $limit: 1,
    },
]);
```

Figura 5 - Consulta que visa apresentar a sala que em média os visistantes demoram mais tempo

Após a conclusão das consultas propostas o grupo constatou as seguintes consultas relevantes:

A seguinte rota tem como objetivo apresentar os 5 productos mais vendidos, como tal, foi necessário primeiramente desconstruir o array de objetos, possibilitando assim agrupar os brindes, após isso, foram ordenados de forma descendente e limitando apenas os 5 primeiros resultados.

Figura 6 - Consulta que visa apresentar os 5 produtos mais vendidos

```
db.visitas.aggregate([
    {
        $unwind: '$brindes',
    },
    {
        $group: {
            _id: '$brindes',
            count: {
                $sum: 1,
            },
        },
    },
    {
        $sort: {
            count: -1,
        },
    },
    {
        $limit: 5,
    },
]);
```

Outra consulta desenvolvida pelo grupo apresenta-nos qual a média de amigos levados por cada visita. Para tal apenas foi necessário agrupar os documentos e efetuar então a média do número de amigos.

```
db.visitas.aggregate([
    {
        $group: {
            _id: null,
            average: {
                $avg: '$numAmigos',
            },
        },
    },
]);
```

Figura 7 – Consulta que visa apresentar a média de amigos levados por visita

Por fim, a última consulta desenvolvida pelo grupo visa apresentar por localidade o número de visitas.

Começando por agrupar por localidade e efetuando o lookup referenciando por id obteve-se o resultado esperado, porém para ficar mais percetível na projeção foi adaptado para mostrar a cidade ao invés do objeto completo.

```
db.visitas.aggregate([
    {
        $group: {
            _id: '$localidade',
            count: { $sum: 1 },
        },
    },
    {
        $lookup: {
            from: 'locations',
            localField: '_id',
            foreignField: '_id',
            as: 'localidade',
        },
    },
    {
        $project: {
            cidade: {
                $arrayElemAt: ['$localidade.city', 0],
            },
            count: 1,
        },
    },
]);
```

Figura 8 - Consulta que visa apresentar a distribuição dos visistantes por localidade

MongoDB Charts

LINK ATLAS - https://charts.mongodb.com/charts-project-0-jyaaw/public/dashboards/61f2a9ec-1d4e-49bd-8109-b14b88f946d7

Para finalizar todo este processo, ficou a faltar a implementação de gráficos na plataforma Mongo Charts. Baseando nas pesquisas anteriormente criadas, realizamos a respetiva implementação visual. Através do MongoDb Charts foi possível de forma fácil e intuitiva mostrar visualmente as consultas criadas. Para tal utilizamos vários tipos de gráficos. Texto, Gráfico de Barras, Heat Map, e Grid.

Utilizamos Widgets texto para apresentar as consultas com valores únicos. Nomeadamente a média de amigos por visita e o número de visitantes com idade superior a 50 anos.

Figura 10 - Média Amigos por Visita

Figura 9 - Número de visitantes com idade maior de 50

Para mostrar o top 5 dos produtos mais vendidos e a média de tempo despendido por sala utilizamos gráficos de barras, tanto verticais como horizontais. Assim podemos mostrar a variável e o respetivo valor.

Figura 12 - Top 5 Produtos mais vendidos

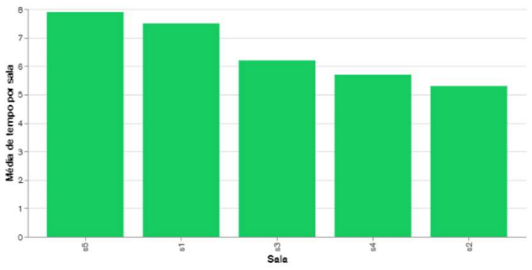

Figura 11 - Média de Tempo despendido por cada sala

Em relação á consulta de distribuição de visitas por distrito, tivemos de recorrer a um gráfico que contemplasse um mapa. Para tal utilizamos um gráfico HeatMap. Assim conseguimos facilmente mostrar a distribuição da origem dos visitantes, em que o valor mais a vermelho são as localidades com maior concentração e a azul com menor concentração.

Figura 13 - Gráfico Heat Map Distribuição

Por fim, foi necessário mostrar visualmente a consulta com mais complexidade. Reparamos que tínhamos 2 variáveis distintas e não seria possível implementar os tipos de gráficos anteriormente utilizados. Sendo assim para sua implementação utilizamos um GRID. Com este tipo de gráfico, é possível mostrar, por localidade e por tipo de produto a contagem do número de vendas. Sendo o mais azul o maior número de vendas e amarelo o menor número de vendas.

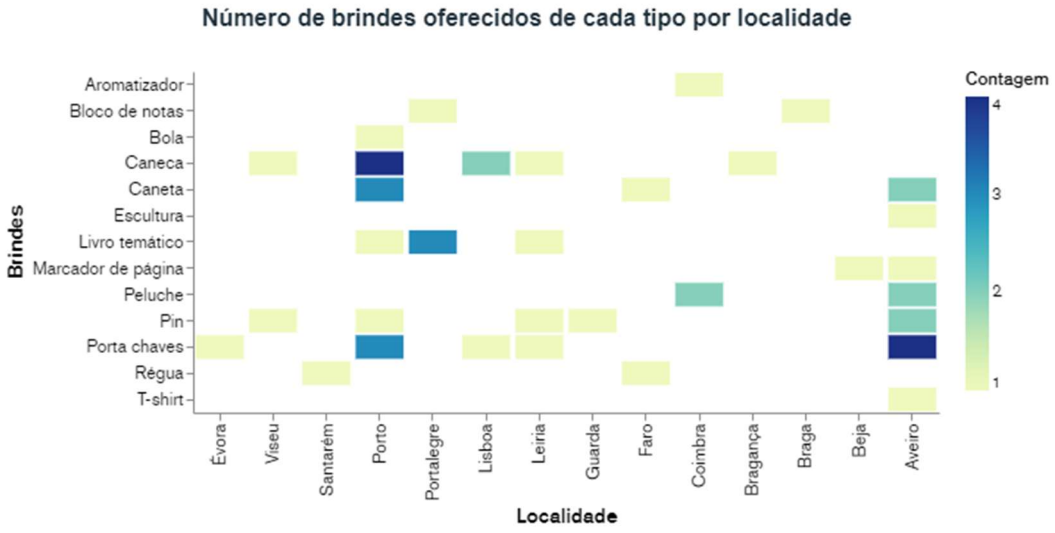

Figura 14 - Grafico GRID Número de brindes oferecidos por localidade e tipo

Conclusão

Após o tema proposto do desenvolvimento de um sistema que permita rastrear o tempo despendido por cada visitante em cada sala do museu os elementos do grupo implementaram para além de funcionalidades básicas, funcionalidades avançadas bem como boas práticas de modo a efetivar o que foi aprendido nas aulas lecionadas para obter o melhor resultado final possível.

Graças á dedicação do grupo, foi possível cumprir todos os objetivos idealizados para a realização deste projeto e solidificar assim ainda mais os nossos conhecimentos acerca do processo, estruturação e integração de dados.

Conforme o exposto, conclui-se que foi um projeto bem conseguido, que do qual estamos orgulhosos de todo o processo de aprendizagem e do resultado final.

Bibliografia

- MongoDB - https://docs.mongodb.com/
- StackOverflow - https://stackoverflow.com/
- Moodle - https://moodle2.estg.ipp.pt/