P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Relatório

Processamento Estruturado de Informação

2021/22

Trabalho prático Época Recurso

Realizado por:

Bruno Dylan Pinto Ferreira, nº 8200586

Gonçalo André Fontes Oliveira, nº 8200595

Jorge Miguel Fernandes Correia, nº 8200592

Nuno de Figueiredo Brito e Castro, nº 8200591

Índice

Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Processamento Estruturado de Informação e consistia em disponibilizar um vocabulário XML e uma API que permitisse aos clientes de uma empresa, a SafeFood uma empresa de segurança alimentar, enviar todos os dados referentes a inspeções, bem como e não conformidades de cada um dos parceiros.

Abordagem

Depois de uma análise detalhada do enunciado decidimos definir uma estrutura que o documento enviado, por cada parceiro, deveria seguir. Posteriormente será explicado todo o tipo de validações realizadas, bem como os requisitos estabelecidos.

De seguida pode ser observado o que cada documento tem de possuir:

Informação sobre o documento (submissionInfo)
Nesta parte do documento, tal como o nome indica, possui dados apenas sobre a submissão do ficheiro, tais como:

- ➔ Data de submissão do documento
- ➔ Semana a que diz respeito a submissão, sendo que é necessário apenas identificar o dia em que começou a semana.
- ➔ Ano de submissão
- ➔ Parceiro que está a realizar a submissão do documento

Informação sobre as Inspeções (inspections)
Aqui serão apresentadas todas as inspeções realizadas sendo que, cada inspeção deverá possuir:

- ➔ Código de inspeção
- ➔ Data em que a inspeção foi realizada
- ➔ Nome do estabelecimento
- ➔ Rating que possuirá um score e um respectivo grade
- ➔ O tipo do serviço
- ➔ Morada do estabelecimento que por sua vez é composto por:

        - Descrição da morada
        - Cidade
        - Estado
        - Código postal
- ➔ Informação sobre o dono do estabelecimento que é composto por:
        - Id
        - Nome
- ➔ Descrição da inspeção
- ➔ Estado do estabelecimento

Uma inspeção possuirá as violações identificadas na mesma onde, cada violação é representada por um código e um estado.

Informação sobre as violações existentes (violationsList)

Por fim, cada documento deverá possuir uma lista de violações existentes na data de submissão do ficheiro onde, cada violação possui o seu código e respetiva descrição.
Esta lista ajudará a que posteriormente, seja possível validar as violações de cada inspeção para que não possam ser associadas às inspeções, violações inexistentes.

Requisitos que limitam o domínio

Depois de uma análise detalhada do enunciado, percebemos que seria necessário estabelecer alguns requisitos para que fosse limitado o domínio dos elementos e assim, assegurar a fidedignidade dos dados. No enunciado eram já apresentados alguns requisitos a estabelecer, no entanto, ao longo da estruturação do problema foram identificadas possíveis requisitos.

De seguida podem ser observados todos os requisitos:

Requisitos comuns:

- ➔ O código de cada inspeção é alfanumérico
- ➔ O código de uma violação tem de possuir 4 caracteres e tem de obrigatoriamente começar pela letra F, tendo de possuir 3 dígitos que se encontram entre 0 e 999(Exemplo: F999).
- ➔ O código postal referente à morada onde foi realizada a inspeção tem de possuir exatamente 5 dígitos.

Requisitos de inspeções:

- ➔ O ***estado de uma violação*** podia tomar apenas os valores:
  - ◆ HOUSING NON-CRITICAL
  - ◆ OUT OF COMPLIANCE
  - ◆ VIOLATION
- ➔ O ***score*** apenas podia variar entre 0 e 100.
- ➔ Uma ***grade*** podia apenas tomar os valores A, B ou C.
- ➔ Um ***serviço*** apenas se trata de:
  - ◆ ROUTINE INSPECTION
  - ◆ OWNER INITIATED ROUTINE INSPECT
- ➔ O ***estado de um espaço*** apenas pode ser ACTIVE ou INACTIVE

Os requisitos apresentados em cima ajudam a manter a uniformidade dos dados para que não sejam causados problemas futuramente em outras partes do projeto.

Propriedades do XML e XML Schema

Depois de feita a identificação do que o documento deveria possuir, bem como, os requisitos que deverá respeitar foram criados os ficheiros XML Schema que podem ser observado de seguida:

- ➔ **CommonTypes:** Possui requisitos que possivelmente podem ser usados em mais que um ficheiro.
  - ◆ Namespace: http://www.trabalhoPEI.pt/CommonTypes
- ➔ **InspectionsRules:** Possui toda a estruturação de uma inspection, bem como todos os respetivos requisitos.
  - ◆ Namespace: http://www.trabalhoPEI.pt/InspectionsRules
- ➔ **ViolationsListRules:** Contém a estruturação que uma lista de violações têm de seguir, bem como os requisitos a serem respeitados.
  - ◆ Namespace: http://www.trabalhoPEI.pt/ViolationsListRules
- ➔ **SubmissionRules:** Possui a estruturação final de um documento, ou seja, uma junção de todas as outras estruturações.
  - ◆ Namespace: http://www.trabalhoPEI.pt/SubmissionRules

Basex

Decidimos que cada documento submetido corresponde a uma submissão, ou seja, uma semana de inspeções realizadas por um parceiro da empresa.
Para isso são submetidos os dados do ficheiro xml no POST request da API, através da ferramenta Postman, utilizando o seguinte endereço 'localhost:8984/addSubmission' ou a partir do endereço 'updateSubmission' quando é feito o update de um ficheiro.

Ao enviar a mensagem no Postman são feitas diversas validações de modo a que a informação guardada não seja comprometida. As validações que são efetuadas são as seguintes:

- ➔ Definimos que a semana começa no domingo e é indicado no enunciado que as submissões são efetuadas no final da semana. Tendo isto em conta, é validado se realmente a submissão acontece 6 dias depois do início da semana, ou seja, ao sábado;
- ➔ Verificamos se o ano indicado na submissão corresponde ao ano da data da semana;
- ➔ É verificado se a data da inspeção pertence à semana indicada na submissão;
- ➔ Para cada inspeção verificamos se as violações que esta contém existem na lista de violações possíveis.

Após serem feitas as validações indicadas, os dados são inseridos na base de dados 'safefood'. Caso a base de dados ainda não exista no Basex, esta é criada.

Para efetuar a alteração de um ficheiro da base de dados, é enviado um novo ficheiro através da API, este ficheiro passa pelas validações indicadas anteriormente e caso passe nas validações, o ficheiro correspondente da base de dados é alterado pelo novo ficheiro.

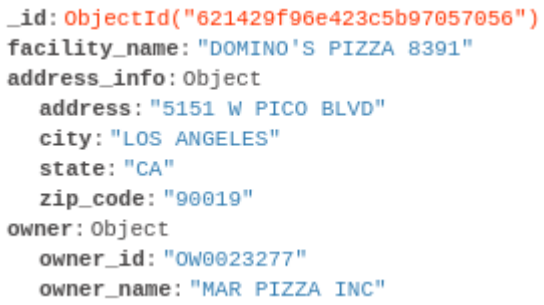
Para exportar os dados da base de dados foi realizado um script de forma a converter o XML para JSON compatível com a estrutura definida no MongoDB.

MongoDB

Estrutura da Base de dados

A base de dados é composta por três collections que são :

1. Facilities - Está collection contém os dados de cada estabelecimento.



Para não haver repetição de dados dissemos que o facility_name e o address_info.city teriam de ser únicos.

2. Inspections - Está collection contém os dados de cada inspeção.

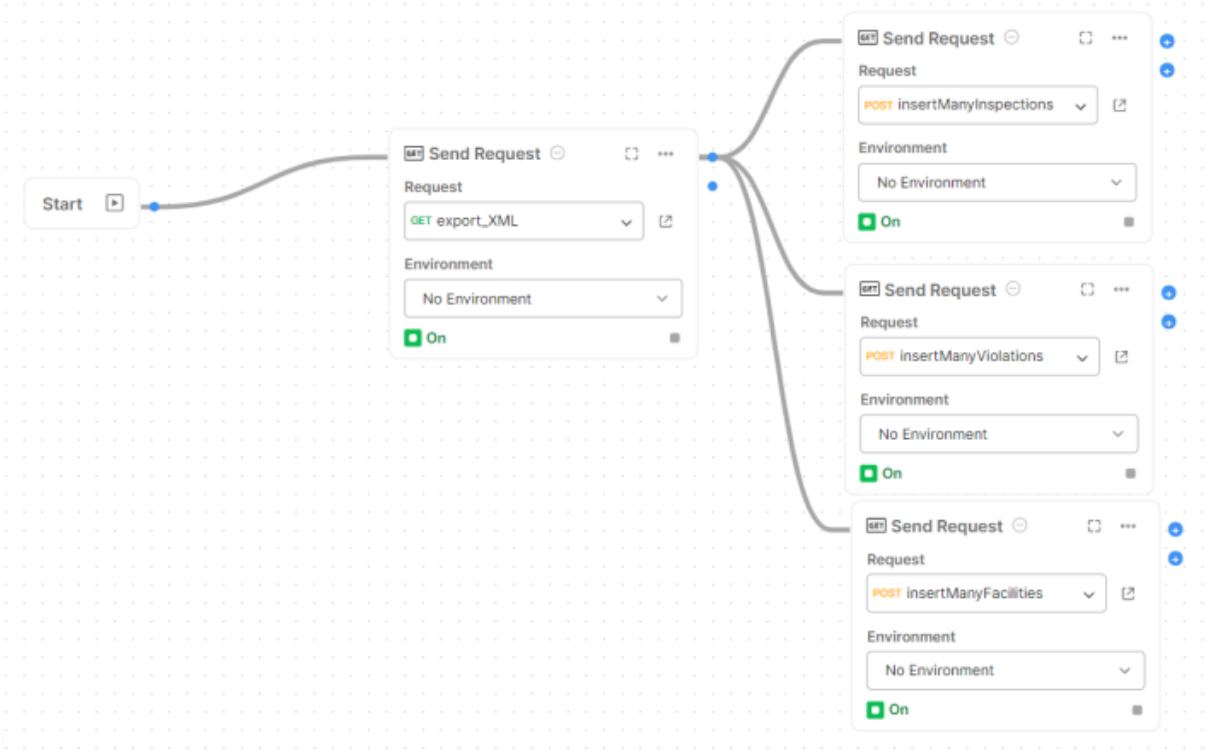
Para não haver repetição de dados dissemos que o code teria de ser único.

3. Violations - Está collection contém todos os tipos de violações.

Para não haver repetição de dados dissemos que o code teria de ser único.

Integração Basex e MongoDB

Para a integração criamos 4 request no postman em que 1 deles recebe todos os dados do basex em xml e transforma em json. De seguida os dados eram enviados para os MongoDB. Para isto criamos a rotina representada abaixo:



Importação dados dos fornecidos no enunciado

Para importar estes dados começamos por criar duas colletions inspections e violations e importar os seus respetivos dados.
De seguida criamos três aggregations que no final da sua execução, criava uma nova colletions com os dados obtidos.
Começamos por criar a aggregations que cria a colletion Facilities, de seguida a que cria a Inspections e por fim a que cria a Violations.

1. Facilities

```
db.inspections.aggregate([{$group: {
 _id: {
  facility_name: '$facility_name',
  city: '$city'
 },
 facility_status: {
  $push: '$facility_status'
 },
 state: {
  $push: '$state'
 },
 zip_code: {
  $push: '$zip_code'
 },
 address: {
  $push: '$address'
 },
 owner_id: {
  $push: '$owner_id'
 },
 owner_name: {
  $push: '$owner_name'
 }
}}, {$addFields: {
 facility_name: '$_id.facility_name',
```

```
 facility_status: {
  $first: '$facility_status'
 },
 address_info: {
  address: {
   $first: '$address'
  },
  city: '$_id.city',
  state: {
   $first: '$state'
  },
  zip_code: {
   $first: '$zip_code'
  }
 },
 owner: {
  owner_id: {
   $first: '$owner_id'
  },
  owner_name: {
   $first: '$owner_name'
  }
 }
}}, {$unset: [
 '_id',
 'facility_status',
 'state',
 'zip_code',
 'address',
 'owner_id',
 'owner_name'
]}, {$out: 'Facilities'}])

db.Facilities.createIndex( { "facility_name": 1, "address_info.city": 1 }, {unique:true} )
```

2. Inspections

```
db.inspections.aggregate([{$addFields: {
 facility: {
  facility_name: '$facility_name',
  state: '$state',
  city: '$city'
 }
}}, {$lookup: {
 from: 'violations',
 localField: 'code',
 foreignField: 'inspection_code',
 as: 'violations'
}}, {$addFields: {
 rating: {
  score: '$score',
  grade: '$grade'
 },
 submission_info: {
  partner: 'Joaquim',
  week: '$date',
  submission_date: '$date'
 }
}}, {$unset: [
 'violations._id',
 'violations.violation_description',
 'violations.inspection_code',
 'facility_name',
 'address',
 'city',
 'state',
 'zip_code',
 'owner_id',
 'owner_name',
 'facility_status',
```

```
 'score',
 'grade'
]}, {$out: 'Inspections'}])

db.Inspections.createIndex( { code: 1 }, {unique:true} )
```

3. Violations

```
db.violations.aggregate([{$group: {
 _id: '$violation_code',
 violation_description: {
  $push: '$violation_description'
 }
}}, {$set: {
 code: '$_id',
 description: {
  $first: '$violation_description'
 }
}}, {$unset: [
 '_id',
 'violation_description'
]}, {$out: 'Violations'}])

db.Violations.createIndex( { code: 1 }, {unique:true} )
```

Mongo Atlas

https://charts.mongodb.com/charts-pei-ucdko/public/dashboards/e4b0d5e6-475c-4e03-b208-f7d63cd0567d

Apreciação crítica

Como foi observado ao longo deste relatório, foi apresentada uma estrutura XML e XML Schema bem organizada que facilita a compreensão da abordagem feita. Além disto, tal como apresentado anteriormente, foram feitas vários tipos de validações para assegurar a validação dos dados que, posteriormente, iriam ser inseridos na base de dados. No entanto, poderiam ter sido realizadas outras validações que não foram realizadas a tempo da data de entrega.

No entanto, podem ser identificadas algumas falhas que não puderam ser resolvidas a tempo de entrega do trabalho, nomeadamente na parte da integração do Basex e MongoDB, quando enviados valores duplicados, juntamente com dados válidos, os dados não são adicionados.

Em suma, apesar de serem visíveis algumas falhas, na nossa opinião achamos que foi realizado um trabalho com qualidade.