

## Implementação em tecnologias *back-end* e *front-end*

Dando prosseguimento ao processo seletivo, gostaríamos que fizesse uma demonstração dos seus conhecimentos técnicos. Para tal, pediremos que faça a implementação de uma *API* e interface *web*, conforme descrito abaixo.

Vamos então aos detalhes!

### 1. API para busca de *Unidades Consumidoras (UCs)*

Uma unidade consumidora (UC), no jargão de uma distribuidora de energia elétrica, é um medidor conectado à rede de distribuição, associado à uma pessoa, física ou jurídica. Usualmente, esse medidor é lido uma vez ao mês, e uma fatura é gerada. Por exemplo, o medidor de sua residência é lido e então você recebe sua conta de luz.

Os clientes da Choice necessitam, comumente, de fazer pesquisas de UCs, por critérios livres como: região, município, bairro, sua classe (ex.: residencial, comercial, industrial), sua atividade (ex.: em classes comerciais, isso representa o tipo de atividade, como uma padaria, um supermercado, um salão de beleza etc).

Implemente uma API REST, com as seguintes assinaturas:

`/ucs`

Método GET retorna a lista completa de UCs.

`/ucs?filter=uc_class eq 'residencial'`

Método GET retorna uma lista UCs, filtradas pelo critério passado na propriedade 'filter'. No exemplo, estão sendo filtradas UCs cuja 'classe' é a residencial.

`/ucs/{id}`

Método GET retorna a UC por seu {id}.

Para os dados, considere baixar o arquivo .zip neste caminho do [Dropbox](#).

O conteúdo deste .zip contém os seguintes arquivos .csv:

UCS.csv – lista com 1 milhão de UCs, com os campos:

- *id* – identificador único da UC
- *consumer\_name* – nome da UC
- *coordinate\_lon* – coordenada geográfica da UC, longitude
- *coordinate\_lat* – coordenada geográfica da UC, latitude
- *id\_region* – identificador da Região onde se encontra a UC
- *id\_locality* – identificador da Localidade onde se encontra a UC
- *id\_municipality* – identificador do Município onde se encontra a UC
- *id\_neighborhood* – identificador do Bairro onde se encontra a UC
- *id\_uc\_meter\_type* – identificador do tipo de medidor da UC
- *id\_uc\_class* – identificador da classe da UC (ex.: residencial, comercial)
- *id\_uc\_phase* – identificador da fase da UC (ex.: monofásico, bifásico)
- *id\_uc\_activity\_type* – identificador da atividade da UC
- *id\_uc\_voltage\_level* – identificador da voltagem da UC (ex.: 110, 220 volts)

Além de UCS.csv, temos os demais arquivos abaixo, que apenas contém os descritores dos campos identificadores (*id\_\**):

- REGIONS.csv
- LOCALITIES.csv
- MUNICIPALITIES.csv
- NEIGHBORHOODS.csv
- UC\_METER\_TYPES.csv
- UC\_CLASSES.csv
- UC\_PHASE.csv
- UC\_ACTIVITY\_FIELDS.csv
- UC\_VOLTAGE\_LEVEL.csv

## 2. Interface Web para a busca de *Unidades Consumidoras (UCs)*

Implemente uma interface web simples, preferencialmente em Angular, que mostre uma lista de UCs, obtida à partir da API feita anteriormente.

Como sugestão, pode ser feito um *grid* paginado, por exemplo:

<div> <div>Search</div> <div>UCClass Name = 'Commercial' ✓</div> <div>☆ 🏠 📄</div> <div>Import UC list</div> <div>Refresh</div> </div>							
UC (Id)	Contract Status	Region	Class	Locality	Municipality	Neighborhood	Phase
100001	Active	Luxembourg District	Commercial	Luxembourg	Luxembourg	Cents	1 Phase
1000015	Cut	Luxembourg District	Commercial	Luxembourg	Hesperange	Hesperange	3 Phases
1000018	Cut	Luxembourg District	Commercial	Esch-sur-Alzette	Kayl	Kayl	3 Phases
1000020	Inactive	Luxembourg District	Commercial	Capellen	Mamer	Mamer	3 Phases
1000059	Active	Luxembourg District	Commercial	Capellen	Kopstal	Kopstal	3 Phases
100006	Active	Diekirch District	Commercial	Diekirch	Mertzig	Mertzig	3 Phases
1000064	Cut	Luxembourg District	Commercial	Luxembourg	Hesperange	Hesperange	1 Phase
1000067	Inactive	Grevenmacher District	Commercial	Echternach	Consdorf	Consdorf	3 Phases
1000081	Active	Diekirch District	Commercial	Clervaux	Parc Hosingen	Parc Hosingen	1 Phase
1000082	Cut	Grevenmacher District	Commercial	Echternach	Echternach	Echternach	3 Phases
1000086	Active	Luxembourg District	Commercial	Luxembourg	Steinsel	Steinsel	3 Phases
1000100	Active	Grevenmacher District	Commercial	Grevenmacher	Biwer	Biwer	1 Phase
1000105	Active	Luxembourg District	Commercial	Mersch	Lintgen	Lintgen	1 Phase
1000109	Active	Diekirch District	Commercial	Vianden	Putscheid	Putscheid	3 Phases
100011	Active	Grevenmacher District	Commercial	Remich	Schengen	Schengen	1 Phase
1000119	Active	Luxembourg District	Commercial	Luxembourg	Weiler - la - Tour	Weiler - la - Tour	3 Phases
1000123	Cut	Grevenmacher District	Commercial	Remich	Stadtbredimus	Stadtbredimus	3 Phases
1000129	Active	Diekirch District	Commercial	Wiltz	Lac de la Haute-Sûre	Lac de la Haute-Sûre	3 Phases
1000131	Active	Luxembourg District	Commercial	Luxembourg	Luxembourg	South Bonnevoie	3 Phases
1000146	Active	Luxembourg District	Commercial	Esch-sur-Alzette	Kayl	Kayl	1 Phase
100015	Active	Grevenmacher District	Commercial	Echternach	Bech	Bech	3 Phases
1000151	Active	Luxembourg District	Commercial	Luxembourg	Luxembourg	South Bonnevoie	1 Phase
1000155	Active	Diekirch District	Commercial	Clervaux	Wincrange	Wincrange	1 Phase

Page 1 of 23921 (598012 items)

<

1

2

3

4

5

...

23921

>

Não se preocupe em ter todas as funcionalidades de um *grid*. Filtros livres completos, validações, ordenações etc. Mas tome nota de como faria 😊

**Notas:**

- A estimativa de tempo gasto nesta tarefa é de 4h a 6h, que você possa fazer nos horários que lhe forem mais convenientes.
- Entendemos que o tempo é escasso para fazer uma aplicação completa. Não se preocupe em ter tudo perfeito e 100%. Pode anotar no código os ajustes que gostaria de implementar mas não haveria tempo, ou mesmo adicione classes e métodos falsos. Vale *pseudo-código* para transmitir a intenção ;-)
- Não é necessário preocupar-se com autenticação ou segurança. Pode deixar API e interface totalmente abertos.
- Pode fazer uso de algum banco de dados ao seu gosto ou, subir o .csv diretamente em memória no .Net. Não se preocupe com persistência ou em manter o banco de dados.
- O volume de dados de teste é grande. Entretanto, caso tenha dificuldades em lidar localmente com esta quantidade, utilize apenas uma parcela dos dados.
- Pode utilizar algum *ORM* como *Entity Framework*, *NHibernate*, *microORMs* etc. Não é mandatório, fica ao seu critério.
- Para a propriedade de filtro, recomendamos, se o tempo permitir, considerar as recomendações da Microsoft, aqui neste [link](#).
- Pode fazer uso de componentes terceiros à vontade, pacotes *nuget* ou *npm*, se assim preferir e facilitar sua implementação. Nós gostamos bastante dos componentes visuais da [DevExpress](#).
- Uma preocupação com arquitetura é bem-vista mas, novamente, entendemos que o tempo é curto para fazer DDD, CQRS etc. Vale citar padrões, referenciar artigos e códigos públicos para expressar-se.
- Codifique como está habituado, com a estrutura de projetos e classes que costuma fazer, nomenclaturas que costuma utilizar. Queremos conhecer o seu estilo :-)

O projeto pode ser feito em Visual Studio 2017 / 2019, Visual Studio Code, tanto faz *.Net core* ou *.Net Framework*, com código entregue como um *.zip* ou repositório público.