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, munícipio, 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 'residential'
   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, latitute
   - *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)

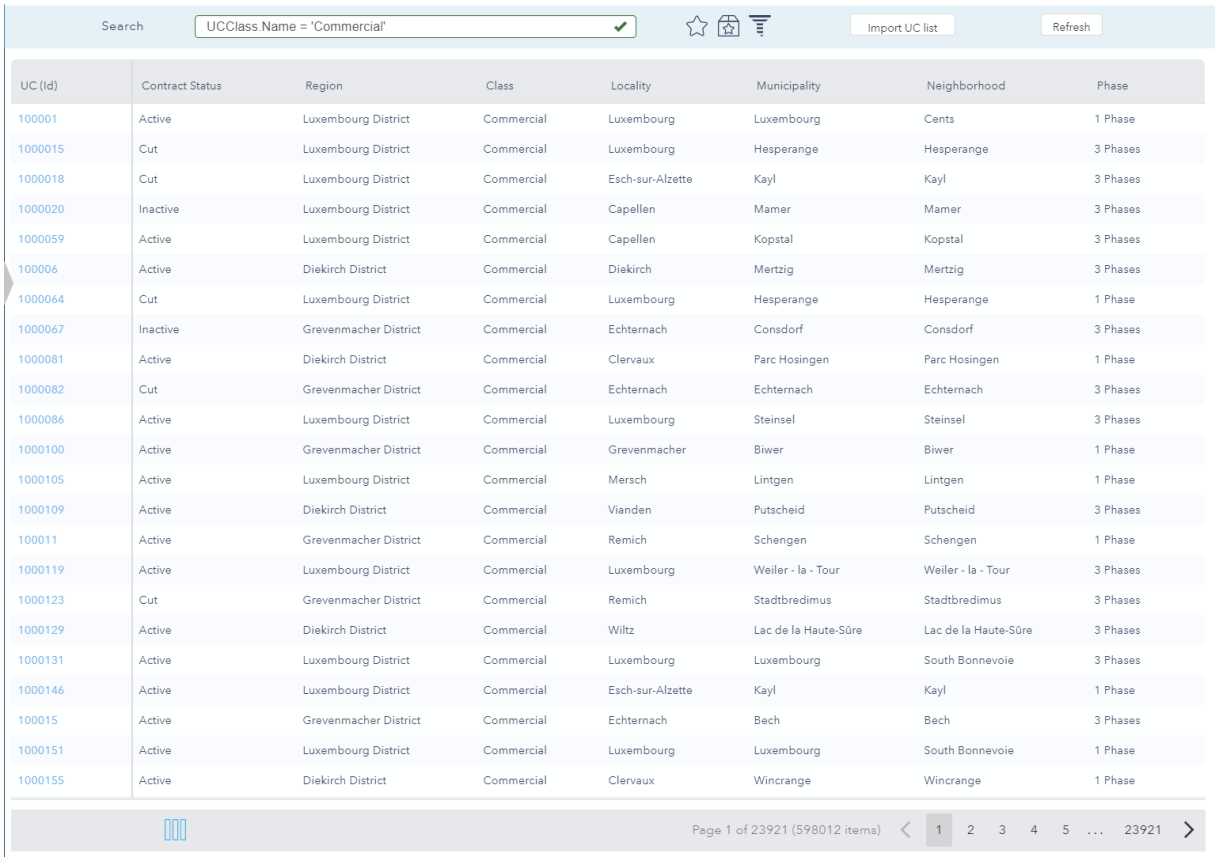
Além de UCS.csv, temos os demais arquivos abaixo, que apenas contém os descritores dos campos identificadores (*id_*):

- 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 inferface 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:



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.