Backend Developer Code Case

Giriş

Bu uygulama sınavında dinamik bir konfigürasyon yapısı kurmanız beklenmektedir. Projenin amacı web.config, app.config, appsettings.json gibi dosyalarda tutulan appkey'lerin ortak ve dinamik bir yapıyla erişilebilir olması ve deployment veya restart, recycle gerektirmeden güncellemelerin yapılabilir olmasıdır.

Konfigürasyon kayıtlarınızı bir storage(MsSql, Redis, Mongo, File vs.) üzerinde tutabilirsiniz. Her bir konfigürasyon kaydı aşağıdaki bilgileri içermelidir.

Id	Name	Type	Value	IsActive	ApplicationName
1	SiteName	string	soty.io	1	SERVICE-A
2	IsBasketEnabled	bool	1	1	SERVICE-B
3	MaxItemCount	Int	50	0	SERVICE-A

Yazacağınız konfigürasyon kütüphanesi(dll), web, wcf, web api her türlü proje için konfigürasyon kayıtlarını ilgili proje için erişilebilir kılmalıdır.

Örneğin **"SERVICE-A"** isimli projemize bu kütüphane'yi eklediğimizde storage üzerinde **"SERVICE-A"**' ya ait tüm kayıtları kendi tipinde dönebilmelidir. Konfigürasyon kayıtları integer, string, double, boolean tiplerinde olabilir. **"SERVICE-A"** aktif olarak çalıştığı sırada konfigürasyona yeni kayıtlar eklenmiş olabilir. Dolayısıyla belli aralıklarla yeni eklenen kayıt olup olmadığı, mevcut bir kaydın değerinin değişip değişmediği kontrol edilerek değerler güncellenmelidir. Konfigürasyon yapısı yalnızca **IsActive=1** olan kayıtları dönmelidir.

Yazacağınız kütüphane en fazla üç adet parametre ile initialize olmalıdır.

```
new ConfigurationReader(applicationName, connectionString, refreshTimerIntervalInMs);
```

ApplicationName: Üzerinde çalışacağı uygulamanın adı.
ConnectionString: Storage bağlantı bilgileri.
RefreshTimerIntervalInMs : Ne kadar aralıklarla storage'ın kontrol edileceği bilgisi.

Diğer taraftan kütüphane'den dışarıya aşağıdaki imza ile bir method açılmalıdır.

```
T GetValue<T>(string key);
```

Örnek kullanım: `_configurationReader.GetValue("SiteName");`

Yukarıdaki gibi bir kullanım söz konusu olduğunda kütüphane sonuç olarak "soty.io" dönmelidir.

Ek olarak bir web ara yüzü ile storage'daki kayıtlar listelenmeli, güncellenebilmeli ve yeni kayıtlar eklenebilmelidir. Kayıtları client side olarak ismi ile filtrelenebilmelidir.

Koşullar

- Kütüphanenin .net 8 ile yazılması gerekmektedir.
- Kütüphane storage'a erişemediğinde son başarılı konfigürasyon kayıtları ile çalışabilmelidir.
- Kütüphane her tipe ait dönüş bilgisini kendi içerisinde halletmelidir.
- Sistem parametrik olarak verilen süre periyodunda yeni kayıtları ve kayıt değişikliklerini kontrol etmelidir.
- Her servis yalnızca kendi konfigürasyon kayıtlarına erişebilmeli, başkasının kayıtlarını görmemelidir.

Ekstra Puan

- Message Broker kullanılması
- TPL, async/await kullanılması
- Olası concurrency problemlerini engelleyecek yapı kurgulanması
- Design & Architectural Pattern'lerin kullanılması
- Gönderilen kod'un TDD yazılması
- Gönderilen kodda Unit testlerin bulunması
- Storage yapısı olarak MongoDb, Redis gibi yapıların kullanılması
- Projenin çalışır halde gönderilmesi
- Proje dokümantasyonu
- Proje kodunun bir source control üzerinden paylaşılması (Github, Bitbucket vs)
- Tüm ekosistemin docker-compose ile çalıştırılabilir olması