1. Geliştirmeye başlamak için tests/provider_test_data klasörünün altına ilgili provider'ı geliştirirken kullanacağınız data bilgilerini ekleyin. (Checkin, checkout, provider_hotel, account type, currency, environment vs.). **Ayrıca ilgili provider'ın ismini tests/base.go dosyasındaki map'e eklemeniz gerekli.**

2. providers klasörünün altına geliştirme yapacağınız provider'ı ekleyip. İlgili provider için adapter, client, credentials, definition, provider_data(gerekliyse) dosyalarını oluşturun.

3. Daha sonrasında ilgili providerın credential kırılımlarını herhangi bir provider credential dosyasını referans alarak tanımlayın.

4. Adapter tarafında geliştireceğiniz 4 tane fonksiyon var. GetClient, GenerateSubqueries, SanitizeRawRequest ve Search.

GetClient: Credential bilgilerini okuyup bu bilgilerle provider client'ını oluşturup döndüğünüz fonksiyon.

GenerateSubqueries: Ana query'yi gerekli koşullara göre subquery'lere parçaladığımız yer. Burada bazı genel kurallara göre baseAdapter'da uygun bölmeyi yapıyoruz. Ancak sizin provider'ınız özelinde farklı bölme senaryoları varsa bunu ayrıca provider adapter'ında uygulamanız gerekiyor.

SanitizeRawRequest: Response'larda gizli kalması gereken credential bilgilerini maskelediğimiz fonksiyon.(Regex)

Search: Provider search endpointi

5. Search request ve response'unu(genel olarak karşılaşabileceğiniz her türlü response formatını) uygun formatlarda struct olarak definitions dosyasına tanımlamanız gerekiyor. Burada provider'ın haberleşme data yönetimine bağlı olarak xml veya json Marshal/UnMarshal kullanmanız gerekecek.

6. Adapter veya Client dosyasında variable tanımlarken generic Adapter veya Client gibi isimler kullanmanız yeterli. Örneğin mikiClient, expediaAdapter seklinde variable isimlerine gerek yok.

7. Provider özelindeki bazı kısıtlamaları baseAdapter'da properties olarak tanımladık. Bunları provider adapter'ında setlemeniz yeterli.

MaxHotelSearchCount : Provider'ın bir search isteğinde maksimum kaç hotel karşılayabileceği. Default değer base'de 500 olarak setlendi.

MaxBookCount: Provider'ın tek transaction'da book isteğinde kaç odayı karşılayabileceği. Şu aşamada bir önem arzetmiyor fakat doğru setlemeniz önemli. Default değeri 1.

MultipleDestinationSearch : Provider aynı anda birden fazla destination ile search yapabiliyor mu ? Default değer true.

ProviderName : Provider ismi.

CurrencyList: Provider'ın desteklediği currency'ler. Default olarak {EUR, USD, GBP}

DestinationSearchThreshold: Bir provederın destinasyon search yapması için ilgili destinasyona ait minimum otel sayısı. Örneğin travcoda bir destinasyona ait 50 ve üzeri otel

oldugunda destinasyon search yapıyoruz. Daha az otel sayısında multiple hotel search yapıyoruz.

MaxNumberofNights: Provider tarafından desteklenen toplam konaklama gün sayısı.

MaxBookingWindowUntilCheckin : Provider'ın desteklediği checkin – today değerinin maksimum değeri.

MaxBookingWindowUntilCheckout: Provider'ın desteklediği checkout – today değerinin maksimum değeri

8. Ayrıca provider özelindeki bazı kısıtlamaları(oda basına maksımum kişi sayısı, maksimum çocuk sayısı vs.) utils/helpers.go dosyasında geliştirdik. Lütfen fonksiyonları yazmadan utils paketine göz atın. Bu pakette kısıtlamalar dışında da birçok farklı işlemi generic fonksiyon haline getirdik.

9. Test scriptini implementation_test [provider_name] olarak çalışıtrmanız gerekiyor. Burada ayrıca pypro flask server'ının da ayakta olması gerekli.

10. Son olarak paketin Go convention'ını koruyabilmek amacıyla repo Readme.Md'de bulunan pre-commit hook'unu local environment'a eklemenizi rica ediyoruz. Bu hook commit atmadan önce ilgili değişiklikleri inceleyerek Go formatına uygun olup olmadığına bakıyor ve format uygun değilse sizin commit atmanızı engelliyor.