Corebit Orientation

Step A : Udemy'den The Complete Node.js Developer Course (2nd Edition)
The Complete Node.js Developer Course (1st Edition) izlenmesi. Ek olarak aşağıdaki linkte faydalı yazılar vs. bulabilirsiniz.

https://medium.com/javascript-scene/composing-software-the-book-f31c77fc3ddc (bu çok önemli bir blog postları serisi, serideki her bir post çok değerli)

https://blog.pragmatists.com/top-10-es6-features-by-example-80ac878794bb
https://flaviocopes.com/javascript-event-loop/
http://exploringjs.com/es6/index.html#toc_ch_arrow-functions

Versiyon kontrol sistemi olarak Git öğrenmek için
https://www.atlassian.com/git/tutorials/comparing-workflows
Ve Node Best Practices
https://github.com/i0natan/nodebestpractices

Step B: Bu çalışmanını amacı promise chaining yapısını kavramaktır. Bu exercise için bir application yazılacak ve aşağıdaki istekler implemente edilecektir. Database yapılarını yazmaya gerek yoktur, applicationdan çağıracağınız bir mock service JSON olarak istediğiniz datayı döndürüp DB'yi simüle edebilir.

Bir satın alma süreci yazacağız ve fonksiyonumuz satın alma işleminin adımlarını gerçekleştirecek. Bunun için purchase(userId, products, options) diye bir servis fonksiyonu yazılacak. Bu fonksiyon içinde user servisinden userı çekerek (getUser) user type kontrolü yapacak ve user type "**kurumsal**" ise firma bilgisi çekilecek (getCompany), bireysel ise devam edilecek. Daha sonra userin seçtiği ürünlerin (products) detayları seçilecek (getProducts). Eğer böyle bir kullanıcı kayıtlı degilse hata dönülecek (hata1).
Her bir ürün için, eğer ürün out of stock ise (product.stockCount eğer 0'a eşitse) user'a hata dönülsün ve işlem bitirilsin. (hata 2)
Her bir ürün için eğer ürün fiyatı 0TL ise, freeOrderTracking tablosuna bu ürün için bir kayıt atılsın ve userId, productId ve işlem tarihi save edilsin.
Sonrasında sipariş tablosuna her bir product için userId, productId, amount ve deliveryAddress kaydedilsin. (delivery Address eğer bireysel user ise user.deliveryAddress bilgisi, kurumsal user ise company.deliveryAddress bilgisinden yararlanarak doldurulsun.)
En son userın emaili varsa user.emailAddress bilgisine sipariş ile ilgili bir "işlem başarılı" emaili at, eger emaili yoksa inactivePurchases tablosuna userId ve productId bilgilerini save et. Böylece emailsiz bir userın sipariş girdiği track edilebilsin.

Promise chaining örneği

```
.then((data) => {
  return dosomething();
})
.then((data2) =>{
return dosomething2();
})
```

.then…..

Step B-2:
Yukardaki çalışmanın aynısını Promise chaining ile değil async/await yapısı ile yazın.

Step C:
Yukardaki örnekte oluşabilecek hata1 ve hata2 için sistemin farklı bilgilerle error throw etmesi istenmektedir. Oluşturulacak hata bilgisinde hata1 için low level hata2 için high level hata dönülmesi istenmektedir. Dolayısıyla purchase() fonksiyonunu çağıran herkes hatanın level bilgisine göre işlem yapabilir hale gelsin.

Step D: Bir express server kurulup, yaratılacak yeni bir endpointte fibonacci hesaplama kodu yazılması ve 300000 sayısı için fibonacci işlemi çalışırken, aynı express servera başka bir istek gönderilmesi ve sistemin davranışının değerlendirilmesi beklenmektedir.

Step E: Native array functionlarının (map, reduce, every, some, filter => ES6 array function) incelenmesi beklenmektedir.
Sonrasında **async (**https://caolan.github.io/async/v3/) kütüphanesinin waterfall,async.parallel,ve async.eachSeries gibi fonksiyonlarının öğrenilmesi gerekmektedir.
Step B'deki exercisede, her bir product için sipariş kaydı girme işlemini async kütüphanesinin each() fonksinunu kullanarak yazın.
Sonrasında ise **ES6** (ecmascript) yapısında getirilen bütün yeniliklerin ve değişikliklerin incelenmesi, en çok kullanışlı olduğu düşünülen 3 tanesin belirtilmesi.

Step F: Javascriptte, code execution standardı olarak promise yapısını kullanmaktayız. Promise kullanarak yazılmış bir fonksiyonu dışardan çağırdığımız zaman, fonksiyon hangi tarih ve saatte execute etmeye başladı, hangi saniyede bitti, fonksiyonun input parametreleri ve return parametrelerinin ne olduğunu loglamayı sağlayacak bir helper fonksiyon yazılması gerekmektedir.

```
const boo = (index) => { }
const foo = (name, desc) => {
        boo(5);
        return 'success';
}
foo('test', 'test2');
```

Diye çağırıldığında aşağıdaki log outputu üretilmelidir.
//foo başladı 10/7/2019 10:39 45 parametreler:test test2
//boo başladi 10/7/2019 10:39 46 parametreler:5
//boo bitti 10/7/2019 10:39 47 return:
//foo bitti 10/7/2019 10:39 47 return: success

Step G: Sequelize öğrenilmesi ve database'e bağlanılması, modellerin yazılması ve kullanılması, sequelize ile sorgu atma kurallarının öğrenilmesi. Tablo ilişkilerinin nasıl yapıldığının öğrenilmesi ve uygulanması

Tablo **Order** tablosu
id:3
name:deneme
orderDate:2017-11-29 15:17:10
company_id:Corebit
müşteri_id:Turkcell

Tablo **orderDetail**
id:5
order_id:3
detailDesc:”abc”
type:sandalye,
price :300

tablo **Company_Parameters**
Id:1
type:username
value:turkcell123
company_id:turkcell

_____
Id:2
type:vergi_no
value:12345
company_Id:digiturk

_____
Id:3
type:fiyat_limiti
value:100
Company_id:turkcell

1.OrderDate e göre now dan 2 gün önceye kadar gelen bütün orderlar.
2.Sandalyesi olan bütün orderları getir (gelen sonuç bir order listesi olacak ve her bir orderın içinde detailleri ile beraber dönecek).
orders:[{ id:3,
company:digiturk,
müsteri_id:turkcell
name:....
Orderdate:...
Details:[{id:5,....type:sandalye},
{id:6…..type:sandalye}
]
{id:4....},
...
]

3.Burada 2no lu exercise'in aynısı, sadece her bir orderın içinde detailleri dönerken, sadece sandalye tipindeki detailleri çeksin).
4. Burada 2nolu exercise'e benzer şekilde, sandalyesi olan orderları değil, **sadece** sandalyesi olan orderları detaylarıyla getirsin. Detayında sandalye ve masası olan order gelmesin.
5. Burada 2nolu exercise'e o firmaya ait username ve fiyat_limiti parametrelerini de ekleyerek liste dönülsün.
6. Burada 5nolu exercise'e ilave olarak vergi no su 1234 olan firmanın bilgileri dönülsün.

Step H: VSCode'a eslint kurarak airbnb-base kural setinin eklenmesi.

Step I: Base Micro Servisin örnek kodlarının incelenmesi.

Step J: Mikroservis yapısı üzerinde basit bir uygulama yazılması. Firmalara ait alertlerin save edilip get edilebileceği bir mekanizma yazılacak. Bir firma admini, userlarına çıkması maksadıyla name, desc, startDate, endDate bilgilerinden oluşan bir alert girebilecek. Çalışanlar ise tek bir endpointten o anda görmesi gereken alertler varsa get edebilecekler. Admin isterse alerti tek bir departmana girebilecek, dolayısıyla sistemdeki kullanıcı, o departmana ait ise o alertin get edilebilmesi gerekmektedir.