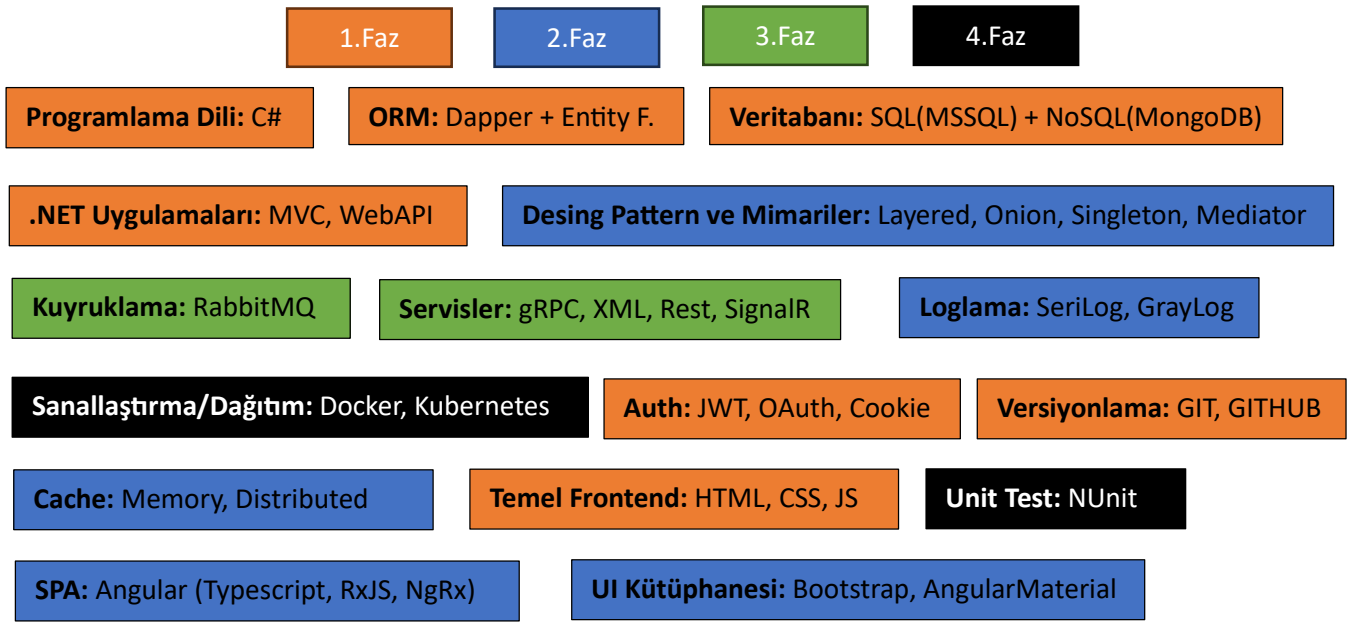
.NET FULLSTACK DEVELOPER YOL HARİTASI

Ben .NET odaklı çalıştığım için kendimce önemli başlıkları ekliyorum. Aşağıdakiler temel programlama bilgisi olan özellikle (C# tarafında) kişiler için hazırlanmıştır. Buradaki başlıkları adınız gibi bilmek zorunda değilsiniz. Kulak dolgunluğu olsa bile kafidir. İhtiyaç halinde açar bakarsınız. Tabii ki ne kadar uygulama yaparsanız o kadar hakim olacağınız için maliyet etkin proje geliştirirsiniz. Bu da sizin diğer adaylara göre seçilme ihtimalinizi ortaya çıkarır. Ek olarak şöyle bir genelleme yapabilirim. Fullstack developerlar genelde backend tarafı ağır basan kişilerdir. Frontend de günü kurtarsak yeter mantığı daha çok ağır basar.

Eğer faz faz olarak düşünürsek aşağıdaki gibi önem derecesi belirlenebilir.



C# HAKKINDA MUTLAKA BİLİNMESİ GEREKENLER

C# Önemli Konular (Udemy-Ücretsiz)
https://www.udemy.com/course/csharp-bilgimi-gelistiriyorum-sorular-ve-cevaplar-ile/

IoC / Dependency Injection (Udemy-Ücretsiz)

https://www.udemy.com/course/aspnet-core-inversion-of-control-ioc-dependency-injection/

LINQ

https://www.youtube.com/watch?v=7N6CXV1yjOE

https://learn.microsoft.com/en-us/dotnet/csharp/linq/standard-query-operators/

Derinlemesine OOP

https://www.youtube.com/watch?v=48Z75_jZHv0&list=PLQVXoXFVVtp306cqgKyC8NoxCmHIuWVBK

Middleware (Exception Handling vs.)

https://www.youtube.com/watch?v=9VeNOtimY3k

Filters

https://www.youtube.com/watch?v=QGCOFV-Za_o

Asenkron İşlemler

https://www.youtube.com/watch?v=IIltT8xqaiI

https://www.youtube.com/watch?v=MxqexDgJlaE

ORM ARAÇLARI

Veritabanı işlemleri için mutlaka en az 1 tane ORM bilmek lazım. Dapper daha hafif raw sql formatında bir ORM tool hızlıca öğrenilebilir. Akabinde çok daha fazla SQL cümlelerinden soyutlayan Entity Framework öğrenilmesi lazım.

(Entity Framework-Hızlıca Aradan Çıksın)

https://www.youtube.com/watch?v=sSJtUQtcONk&list=PLRp4oRsit1bwXZy15yCJJdpN5ESGGzAC5

(Entity Framework-Derinlere)

https://www.youtube.com/watch?v=dbI-kostQWo&list=PLQVXoXFVVtp1o3nq3-IXv42bPaFlzroBE

VERİTABANI UYGULAMALARI

SQL için oldukça fazla tekrar etmek lazım. Örnek bir veritabanı üzerinden şuradan şu veriyi nasıl alabiliriz, iki tabloyu birleştirip nasıl verileri topları vs. gibi konular için kendi kendine senaryolar üretmek lazım.

https://www.btkakademi.gov.tr/portal/course/tsql-ile-veri-tabani-programlama-22791

https://www.btkakademi.gov.tr/portal/course/uygulamalarla-sql-ogreniyorum-8249

https://www.btkakademi.gov.tr/portal/course/tsql-ile-veri-tabani-programlama-22791

Stored Producerler, Viewlar, iki tabloyu birleşitirip sanal tablo oluşturma oradan verileri alma gibi işlemler mutlaka bilinmesi gerekiyor.

Senaryo bazlı örnekler çözmek için ücretli bir Udemy eğitimi var.

https://www.udemy.com/course/alistirmalarla-sql-ogreniyorum/

İlişkisel olmayan doküman odaklı diyebileceğimiz NoSQL veritabanı olan MongoDB hakkında derinlemesine olmasa da bilgi sahibi olmak lazım.

MongoDB Temel Eğitimi

https://www.youtube.com/watch?v=_hb9fnWNHrk&list=PL0BR3UnhlDq6vS6u1eOjCRCBRS7AbufVs&index=7

.NET UYGULAMALARI(MVC-API)

.NET Web API Eğitimi

https://www.youtube.com/watch?v=XI1HYLUqNnI&list=PLTLwdny-C3ttHmkuXtDBtaYF0P32vcakI

.NET MVC + Web API Eğitimi

https://www.youtube.com/watch?v=PhHEn17ms9o&list=PLTLwdny-C3tu2qbPQFpy7JSrHHQ7Lnjzn

Derinlemesine .Net Core Eğitimi

https://www.youtube.com/watch?v=RMEhZjnoTrY&list=PLQVXoXFVVtp33KHoTkWklAo72l5bcjPVL

DESİNG PATTERN VE MİMARİLER:

Bu konular çok soyut olduğu için öğrenmesi zor. Ancak zaten internet üzerinden yapılan birçok uygulamada farkında olunmadan tasarım kalıpları ve mimariler uygulanıyor. En çok bilinmesi gereken mimari (proje yapısı / structure) zaten katmanlı mimaridir. Son zamanlarda API'lar ile birlikte Onion mimaride ön plana çıkıyor. Bununla birlikte kullanılan CQRS design patternı da önplana çıkıyor.

Ama design patternlar uygulama geliştirmede senior seviyedeki biri için gereklidir. Mevcut uygulamanın hangi pattern üzerinde olduğu aktarılması veya anlaşılması sonrasında gözde büyütülecek bir konu değildir. Eğer sıfırdan bir proje yapılacaksa ve ihtiyaçlar belirlenmişse burada hangi mimari ve tasarım kalıbı kullanılacağını takım lideri / senior developer karar verir.

Şimdilik katmanlı mimari bilinse bile yeterli olur.

Desing Patterns ve Architectures

https://refactoring.guru/design-patterns/catalog

https://www.turing.com/blog/software-architecture-patterns-types/

Mesela yukarıdaki bağlantılardaki başlıkları karşılaştırmaları vakit oldukça okumakta fayda var. Bunları birebir uygulamaya çalışmak şimdilik zaman kaybıdır. Başlangıçta dediğim gibi katmanlı mimari/onion architecture bilmek fazlasıyla yeter. Diğerleri hakkında genel kültür olsa kafidir.

.NTier Architecture (Layered Architecture)

https://www.youtube.com/watch?v=jmS8mz_KAVo&list=PLDSvesNxEuJOpy7_TuTYqfMWFmEaYJqSe

-Baştan Sona Uygulama

https://www.youtube.com/watch?v=Lpo0avv3g-Q&list=PLrSCwxkucNmxFrrAsGm14Z-5Cu52MKrNr

Onion Architecture

https://www.youtube.com/watch?v=Q1XyDTmm4tw

https://www.youtube.com/watch?v=CCWTITXALGo

https://www.youtube.com/watch?v=GDKy2xZsZhs

-Baştan Sona Uygulama

https://www.youtube.com/watch?v=luTUl8CSudM&list=PLrSCwxkucNmw_sjxZZHaWj6ySakPgSCbv

KUYRUKLAMA

RabbitMQ

Yine bu da derinlemesine bilinecek bir konu değil. Sadece ne işe yaradığının bilinmesi kafidir. Önüne bir sorun çıktığında bunu kuyruklama mekanizması ile çözebilirim demek yeterlidir.

https://www.youtube.com/watch?v=QGHP8Bi5Q5E

LOGLAMA

Projelerin olmazsa olmaz ihtiyaçlarından biride loglamadır. Bunun için birçok yöntem kullanılabilir. Proje ihtiyaçlarına göre belirlenmesi gerekiyor.

SeriLog

https://www.youtube.com/watch?v=RcmdLfcFduE

GrayLog

https://www.youtube.com/watch?v=5X3jsV2EGbE

SERVİSLER

Restful / Restful API

https://www.youtube.com/watch?v=2s2dp3bxWhI&list=PLr48dQTh3FFzwDwCvkVt8Mjxhfjnw0Scf&index=1

https://www.youtube.com/watch?v=GGUq9MA2JIs

gRPC

https://www.youtube.com/watch?v=FFqg-WhhOw4&list=PLQVXoXFVVtp3oS21qi7a0DZikNPAWxevZ

SignalR

https://www.youtube.com/watch?v=hIW3wt3tvmc&list=PLQVXoXFVVtp3RSycdru4WpnfPEOFxONiX

XML (WSDL)

https://www.youtube.com/watch?v=3oRBebVBpJM

CACHİNG

Projede ihtiyaç olursa ele alınabilecek bir şey. Yani ne zaman kullanmak gerektiğini, avantaj/dezavantajları bilsek yeter.

Memory Cache

https://www.youtube.com/watch?v=aFpOSfm3F64

Distributed Cache (Redis)

https://www.youtube.com/watch?v=JLS9gg-oJPQ

https://www.youtube.com/watch?v=SoH5x8dk6iM

VERSİYONLAMA

https://www.youtube.com/watch?v=-kYHuFtCX7A&list=PLv1CRNciwsrcFph511nrrdAbY5EYV89uk

Genelde GIT kodlarını kullanmıyoruz. Github Desktop üzerinden işlemleri hallediyoruz. Ama genel bilgi sahibi olmakta fayda var. Özellikle, merge işlemleri, branch çıkma, reset, revert, stash, hatta cherry pick konusunu da öğrenmekte fayda var. Zaten belirli bir noktadan sonra otomatize oluyor kafada büyütülecek konu değil.

AUTHENTICATION

Authorize ve Authentication konuları olmazsa olmazlar arasındadır. Authentication işlemleri için en popüler olan JWT kullanımıdır. Authorize mekanizmasını ise proje içerisinde kurgulama lazım. Projeden projeye değişir.

Cookie

https://www.youtube.com/watch?v=NI5h0YiaMhs

JWT

https://www.youtube.com/watch?v=lCcz9N6-B7Q

https://www.youtube.com/watch?v=ju-53ZyjfEA (JWT ile güvenlik konusu genel kültür)

OAuth

https://www.youtube.com/watch?v=gq1zNiKWCP4

SANALLAŞTIRMA/DAĞITIM

Bu konu aslında orta ve büyük ölçekli şirketlerde birebir developer sorumluluğunda değildir. Bunu devops ekibi ele alır. Ancak fikir sahibi olmakta fayda var. Zaten küçük ölçekteki bir firmada da devops konusu pek önemsenmez, sanallaştırma/dağıtım konusu elzem değildir.

Docker

https://www.youtube.com/watch?v=uk1EIye81uY&list=PLRp4oRsit1bzGGClDYCplnGKYI6p-dDE1

Kubernetes

https://www.youtube.com/watch?v=Ca7AebYNaKA&list=PLViWvmuLtSyPuoxiJRjmLo0t-cdfSZZON

https://www.youtube.com/watch?v=civATsL0MCk&list=PLm8ggkC19szD2p32g4VLDhUVnJtfwubxl&index=2

TEMEL FRONTEND

Aslında bir fullstack developer için Frontend tarafı hep geri planda kalmaktadır. Çünkü bu tarafta bir derya deniz. Bu nedenle hepsinde iyi derecede bilmek mümkün değil. Çok uçanın kaçanın olmadığı temel/orta düzey bir Frontend bilgisi yeterli olacaktır. Zaten .Net ekosistemi tarafından ilerleyen bir developer için eğer SEO kaygısı güdülmüyorsa Angular gibi bir SPA kullanılarak ilerleniyor. Ya da klasik asp.net mvc projesi üzerinde ön yüzde Bootstrap kullanılarak yola devam ediliyor.

Mesela bir frontendci daha fazla UI kütüphanesi / aracı bilir. NodeJS tarafında tool kullanımı bilir. Daha spesifik işler (animasyonlar vs.) yapabilir. Ancak bir fullstack developerdan bunlar beklenemez. Mesela pagespeed metriklerine göre kullanılan araçların özelleştirilmesi gibi konularda bir frontendci daha iyi olmak zorundadır.

HTML & CSS & JS & JQuery

https://www.youtube.com/watch?v=uHEr6d6EftA&list=PLURN6mxdcwL_D8H1iki2YCmp-lNyNAdbz&index=2

ANGULAR

.NET geliştiricileri daha çok Angular tarafına yatkındır. Temelinde Typescript kullandığı için kendilerine daha yakın gelmektedir. Ayrıca react gibi bir çok harici paketi bir araya getirmek yerine kendi içerisinde dahili olarak bir çok yapı geldiği (örn: routing) için daha stabildir. Ancak unutulmamalıdır ki kendi içerisinde bu da bir derya denizdir. Bu konuda şanlıyız ki çok iyi bir Angular eğitimi var.

https://www.youtube.com/watch?v=Z4WqBwmO0mA&list=PLQVXoXFVVtp1DcC4z0euk71_ICphrOEFV

Temel düzeyde bu yeterli olsa da Angular tarafında daha temiz işler yapmak için (işte burada bir Frontend dev daha çok şey biliyordur) Typescript, NgRX (state management), RxJS (reaktif programlama) bilmek gerekiyor. Yine de bunları gözde büyütmemek lazım. Temel mantığını bildikten sonra ihtiyaç halinde aç dokümanı oku diyebiliriz.

RxJS

https://www.youtube.com/watch?v=tEUda4YzCI4&list=PLQVXoXFVVtp1v1_D_8ocGOsWFGvK1Ha-E

TypeScript

https://www.youtube.com/watch?v=WdcZE4DkOuE

NgRx

https://www.youtube.com/watch?v=0bt4kQlsKgA

https://www.youtube.com/watch?v=YRgux3zKwNQ&list=PLLbsh38WYXTGDjIkGofDqsySslScFTwMT

UI KÜTÜPHANESİ

UI kütüphaneleri işlerinizi kolaylaştırmak için vardır. Bir projede eğer çok spesifik şeyler istenmiyorsa yani işin mühendislik kısmına girilmeyecekse UI kütüphaneleri ihtiyaçlarınızı karşılayacaktır.

Bootstrap

https://www.youtube.com/watch?v=1RsTYXreso4&list=PLK-Y7MqjKK0BvU8_lkIX2QEDtCC4vltop&index=12

(Derinlemesine)

https://www.youtube.com/watch?v=3D7gVylUglw&list=PLY20HpFruiK13LB2jTkET4DkJP9eD2HZf&index=3

Angular Material

Angular için yerleşik olarak desteklenen bir UI kütüphanesidir. Angular component mantığını doğrudan kullanabildiğiniz (direktifler vs.) için Angular geliştiricileri için inanılmaz kolaylık sağlayacaktır. Aslında açıp dokümandan da ihtiyaç halinde takip edilebilir illa bir eğitime gerek yok.

https://www.youtube.com/watch?v=NWfldvi3Yhs&list=PL3VYzfbLCSS8U2BCvV7HpyObxM9g1Y7Xd&index=5

NOT: Angular tarafında kullanmak üzere ngBootstrap kütüphanesi de bulunmaktadır. Bildiğiniz Bootstrap kütüphanesinin Angular'a uyarlanmış halidir.

https://ng-bootstrap.github.io/#/home

UNIT TESTING

Test konusu çok sevilmeyen ve boşa vakit kaybı gibi görünen bir konu ancak kesinlikle fikir sahibi olmak lazım. Test yazmasanız da (takımda tester vardır mesela) test edilebilir kod yazmalısınız. Unit test aslında bir servisi veya bir metodu test etmek için kullanılır. Başka bir yazılımcı geldiğinde ilgili servise kod yazdığın daha önceden belirlenmiş senaryoların herhangi birinde probleme sebep oluyorsa canlıya çıkmadan fark edilmesini sağlar.

NUnit

https://www.youtube.com/watch?v=Qe0DFbzJ1WI&list=PLRp4oRsit1bxg1Dp6PnyGL6YXSw3-4Jbh&index=1