**HAZIRLAYANIN ;**

**ADI :** EMRE

**SOYADI :** YAPICI

**BRANŞI :** BİLGİSAYAR MÜHENDİSİ

**DEPARTMANI**: ARGE TAKIMI

**GÖREVİ :** STAJYER

**KONULAR:** FRONT-END, BACK-END, MIDDLEWARE, SOFTWARE LIFECYCLE, REST API ve SOAP API ,DATABASES, DB FIRST, CODE FIRST, MODEL FIRST, GIT, GITHUB

**FRONT-END NEDİR VE TEKNOLOJİLERİ NELERDİR ?**

**Front-End** web sitelerindeki gözüken bütün fonksiyonların, sistemsel kısımları (menü, search alanı ..vs) kısımların çalışmasını sağlayan web sitesi geliştiricilerdir.

HTML, CSS , Javascript , Python , java, react gibi programlama dilleri kullanılabilir.

Web tasarımı için bilinmesi gereken teknolojiler:

Ecmascript, api, Restful api, digital ocean, docker, nodejs, Express js, npm yarn ve sketch.

**JSON yapıları**:

JSON açılımı JavaScript Object Notation. Yapısı XML'e benzer. XML'e göre daha hızlıdır

key-value çiftlerinden oluşur. Key string tipinde value ise her tip olabilir (integer , char , string , float…).

**BACK-END NEDİR VE TEKNOLOJİLERİ NELERDİR ?**

**Back-End**; sunucu ile kullanıcı arasında iletişimi sağlayan alandır.

Başlıca **Back-End** dilleri:

PHP , NodeJs, Py , C# , GoLang , Ruby , Java, Kotlin .

**Backend FrameWorkleri** şunlardır:

NodeJs (Express, Koa, Nest, Loopback), py (Django, Flask), PHP (Laravel, Symfony, Slim, Codigniter), C# (ASP .NET), Java (Spring MVC), Ruby (Ruby onRails, Sinatra), Kotlin (Javalin, KTor).

**Challange** siteleri:

Leetcode, Hackerrank, Coderbyte

**VeriTabanları:**

PostressSQL ,MongoDB , MySQL, MSSQL Server, Firebase Elasticsearch

Uzmanlaşmak için bazı **teknolojiler.**

Hosting Platformları: Heroku, Digital Ocean, AWS, Azure, Netlify .

Web Sunucu: Nginx, Apache.

Konteynerler: Docker, Kubernates.

**MIDDLEWARE NEDİR ?**

**Middleware**, birbirleri ile iletişim halindeki uygulamalar arasında, tekrar eden görevleri gerçekleştirmek amacıyla yer alan ara yazılımlardır.

Tek başlarına bir işlem gerçekleştirmezler.

Ancak, doğrudan bir iletişim başlatmazlar ya da veri alışverişi gerçekleştiremezler.

**Middleware,** birbirleri ile iletişim halindeki uygulamalar arasında, tekrar eden görevleri gerçekleştirmek amacıyla yer alan ara yazılımlardır.

Uygulamaların iletişiminde SOAP, REST ve JSON gibi yapılar kullanılabilmektedir.

Ara seviyelerde (level) konumlandırılabilir:

Uygulama düzeyinde ara katman yazılımı,

Yönlendirici düzeyinde ara katman yazılımı,

Hata işleme ara katman yazılımı,

Yerleşik ara katman yazılımı,

Üçüncü taraf ara katman yazılımı

**REST API VE SOAP API**

**Rest**, istemci-sunucu arasında hızlı ve kolay şekilde iletişim kurulmasını sağlayan bir servis yapısıdır.

XML veya JSON verilerini taşıyarak uygulamaların haberleşmesini sağlar.

**Soap**  en temel anlamda, internet üzerinden küçük miktarda bilgileri yada mesajları aktarma protokoludur

XML tabanlı kullanıma mecbur bırakır. Bu konuda esnek değildir.

**Rest ile Soap Farkı**

Güvenlik: SOAP güvenlik sağlamak daha kolay ve hızlı şekilde REST güvenlik söz konusu olduğunda SOAP’a göre biraz zayıf kalmaktadır.

Data Yapısı: REST ile JSON, XML hatta TEXT ile çalışabiliyorken SOAP ile XML kullanmalıyız. REST daha kullanışlı olabiliyor.

Uygulama Hızı: Uygulamanızın daha hızlı çalışmasını istiyorsanız REST kullanmanız yararınıza olacaktır.

Destek: SOAP için geliştirici araçları daha iyidir.

Kullanmamız için bize yardımcı olacak daha çok kaynak bulabiliriz.

REST’in dökümantasyonu SOAP’a göre daha azdır.

**XML nedir ?**

İnterneti kullanarak veri alışverişi yapan sistemler ve platformlar arasındaki veri iletişimini

standart hale getirmek için tasarlanan bir işaretleme dilidir. Verileri belli bir yapıda derleyip, başkalarının bu verileri kullanmasına imkan tanımasıdır. İsimlendirme konusunda insiyatifi bize bırakan XML, diğer konularda katı ve kesin kurallara sahiptir.

Bir XML dosyası oluşturulurken aşağıdaki kurallara uyulması gerekir.

· XML etiketleri hiyerarşik bir yapıda oluşturulmalıdır.

· Açılan etiketler mutlaka kapatılmalıdır.

· XML’ de etiketlere nitelik atanabilir.

· Xml Case Sensitive bir dildir.

**Json ile Xml farkı nedir?**

JSON; okunması ve kullanımı kolay XML benzeri içerik kategorileme dilidir.

Aslında XML dilinin veri alış verişi sırasında büyük ve yavaş kalması üzerine geliştirilmiştir

JSON, XML’e göre daha pratik ve daha hızlıdır. XML gibi bir çok programlama dilinde kullanılabilir.

**SOFTWARE LIFECYCLE (YAZILIMSAL YAŞAM DÖNGÜSÜ)**

Yazılım yaşam döngüsü, yazılımın hem üretim hem de müşteri tarafından kullanımı süreçlerinde

devam eden yazılımı geliştirmek için geçirdiği aşamaların tümüne verilen isimdir diyebiliriz.

Yazılım yaşam döngüsünün temel adımlarını genel olarak aşağıdaki gibi sınıflandırabiliriz:

**Planlama (Planning) :** müşterinin gereksinimleri alınır ve planlar yapılır.

**Analiz (Analysis)** : çözümün incelenip gereksinimler incelenir ve projede analiz çalışmaları yapılır.

**Tasarım (Design) :** Mantıksal Tasarım: Önerilen sistemin yapısını anlatır. Olası örgütsel değişiklere önerilir.

**Fiziksel Tasarım:** Yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.

**Gerçekleştirme (Implementation) :** Kodlama ve test aşamaları yapıldıktan sonra kurulumun yapıldığı aşamadır.

**Bakım (Maintenance) :** proje için varsa hatalar düzeltilir, iyileştirmeler yapılır, ürün için istenilirse yeni özellikler eklenir.

**VERİ TABANLARI (DATABASES)**

Veri tabanı, yani database ise düzenli bir şekilde toplanmış veriye verilen isimdir.

Burada düzenli kelimesine dikkat çekmek gerekir, veri tabanındaki verilerin kolayca ulaşılabilir ve yönetilebilir olması gerekir.

**SQL**

SQL neredeyse bütün veri tabanlarında sorgulama, üzerinde oynama, veri tanımlama ve erişim kontrolü sağlama için kullanılan bir programlama dilidir.

Dosya tabanlı: veriler düz bir dosya içinde muhafaza edilmiştir.

İlişkisel veri tabanları: Kolonlar ve sıralar hâlinde tablolar içinde düzenlenen veriler içerir.

Nesne odaklı veri tabanları: Nesne odaklı veri tabanlarında veriler nesne formunda sunulur.

Bu hâliyle nesne odaklı programlamaya benzer.

Dağıtılmış veri tabanı: İki ya da daha fazla dosyanın farklı sitelere dağıtılmış olduğu veri tabanlarına verilen isimdir. çoklu bilgisayarlarda muhafaza edilebilir, cihazlar aynı mekânda ya da

farklı ağlara sahip ortamlarda bulunabilir.

Veri ambarları:Hızlı sorgulama ve analiz için ideal veri tabanlarıdır.

**NoSQL**

NoSQL veri tabanları:NoSQL veri tabanlarına ilişkisel olmayan veri tabanları da denir. Düzenlenmemiş ya da yarı düzenlenmiş verilerin muhafaza edilmesi ya da üzerinde oynanması için ideal veri tabanlarıdır

Açık kaynaklı veri tabanları: kaynak kodu açık kaynaklı kod olan veri tabanlarıdır

Bulut veri tabanları: Yöneticisi verilerin özel, topluma açık ya da hibrit olmasını tercih edebilir.

Kullanıcıya esneklik sağlayan bulut veri tabanlarının iki ayrı modeli bulunmaktadır

Birincisi geleneksel bulut veri tabanıyken ikincisi bir hizmet olarak veri tabanıdır (DBaaS). DBaaS’lar

hizmet sağlayıcı olarak yönetim ve bakım işlemlerine izin verir.

Veri tabanı yazılımı, veri tabanı dosyaları ve kayıtları yaratmak, bunları düzenlemek ve bakımlarını yapmak için kullanılan yazılım programları ya da araçlarıdır.

DBMS’ler ayrıca veri tabanlarının kontrol edilmesi, ayarlarının yapılması yedeklenmesi ve beklenmedik durumlarda kurtarılması için de kullanılır.

en popülerleri:

MySQL, Microsoft Access, Microsoft SQL Server, Oracle Database, Dbase, Firebird, FileMaker, MS Access

**Db First , Model First, Code First**

Database First, mevcut veritabanından bir modele ters mühendislik uygulamanıza olanak tanır.

Database first yaklaşımından yararlanabileceğimiz bazı durumlar;

Az değişen şemalarla, uzun ömürlü veritabanlarında kullanılabilir.

Modelinizi bir diyagramda görmek ve buradan güncellemek istiyorsanız kullanılabilir.

Modelinizi çakışmayan diyagramlara bölmek isterseniz kullanılabilir.

DBA’lar tarafından tasarlanmış bağımsız geliştirilen bir veritabanınız var ise kullanılabilir.

Sfırdan bir veritabanı oluşturarak projeye başlanması gereken durumlar için ideal olabilmektedir. Kod yazmadan ya da serverda fiziksel database ile ilgilenmeden, tamamen entity model üzerinden veritabanı oluşturmak için kullanılan yaklaşımdır.

Model First:

Entity user interfacesi üzerinden modellediğiniz verilerinizi generate ettiğinizde hem database tarafında hem de kod tarafında otomatik olarak oluşurlar.

Code First, adından da anlaşılacağı üzere kod ile veritabanı ve entity modeli tasarlama

yaklaşımıdır. Yapmanız gereken tek şey kodlarla entity classlarını tanımlamak olacaktır.

**Git ve GitHub**

Projemizde bir değişiklik veya bir yenilik yapmak istiyoruz direkt olarak projemizde değişiklik yapmak yerine kopyasında değişiklik yaptığımızı düşünün böyle yaptığımız zaman bir hata ile karşılaştığımızda ilk sürüm olan orijinal projenize dönüş yapabiliriz. İşte **Git** tam olarak bunu kolayca yapabilmemizi sağlıyor.

Temel Git Komutları

"git config" Komutu:

Bu komut sayesinde bilgisayarınızdaki git terminalini kendi Github hesabınıza bağlayabilirsiniz.

git config –global user.name “kullanıcı adınız”

git config –global user.email “kullanıcı e-mail adresiniz”

Yukarıdaki iki komutu terminale girdikten sonra, terminalde yapacağınız her işlem artık Github hesabınızda da uygulanacaktır.

"git init" Komutu:

Git init komutu bilgisayarınızda oluşturmuş olduğunuz bir klasörün Git'e tanımlanması için kullanılmaktadır. Yani, masaüstünüzde oluşturduğunuz bir klasörü yerel bir repository (depo) yapmaya yarar.

"git clone" Komutu:

"git clone" komutunu kullanarak Github'ta bulunan bir repository'i bilgisayarınıza kopyalayabilirsiniz.

"git add" Komutu:

Projenizde yaptığınız değişiklikleri çalışma dizinine (repository) ekler. Kısacası commit etmeye hazır hale getirir.

"git push" Komutu:

Oluşturmuş olduğunuz projeyi Github'taki kaynağa gönderir.

"git pull" Komutu:

Proje ana dosyasındaki yaptığınız değişikliklerin bilgisayarınızdaki versiyonuna çekilmesini sağlar.

"git commit" Komutu:

"git add" komutu ile eklediğiniz tüm değişiklikleri kayıt etmek istiyorsanız "git commit" komutunu kullanabilirsiniz. Aşağıdaki komut, dosyanızda yapılan değişiklikleri kaydetmenize yarar.

git commit -m "Açıklama"

"git status" Komutu:

"git status" komutu kısaca repository hakkında bilgi verir.

"git branch" Komutu:

Bu komutu kullanarak tüm branchleri görebilirsiniz.

"git merge" Komutu:

Üzerinde çalışmış olduğunuz iki parçayı birleştirme işlemini gerçekleştirir.

**GitHub** servisi sayesinde pek çok kişiden oluşan kalabalık yazılım geliştirici ekipler bile aynı anda aynı proje üzerinde çalışabilir ve ayrı değişiklikler yapabilirler. Bu servis üzerinden geliştirilen

projelerde ekip büyük bir zaman kazanır.

Çakışmalar olmaz ve uzun toplantılarda karar verilecek şeyler belki bu servis üzerinden gösterilerek hızla onaylanabilir.

**KAYNAKÇA (REFERENCES) :**

<https://medium.com/@tunaytoksoz/yazilim-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-sdlc-ve-modelleri%CC%87-c3fe40f6e4e8>

<https://berqnet.com/blog/veri-tabani-database>

<https://ciftci-oguzhan.medium.com/net-core-database-first-kullan%C4%B1m%C4%B1-53eb488f1939>

<http://cagataykiziltan.net/entity-framework-database-first-model-first-code-first-yaklasimlari/>

<https://www.soltrabilisim.com.tr/entity-framework-code-first-avantajlar/>

<https://coderspace.io/blog/git-nedir-nasil-kullanilir/>

<https://www.hostinger.web.tr/rehberler/github-nedir>

<https://www.webtekno.com/github-nedir-ne-ise-yarar-h117655.html>

<https://medium.com/>