Web ve Bulut Geliştirmeye Genel Bakış

Web Tarayıcıları ve İletişim

- İnternet tarayıcıları (örneğin, Google Chrome, Mozilla Firefox) URL'ler aracılığıyla sunucularla iletişim kurar.
- Sunucu, HTML, CSS ve JavaScript gibi verileri döndürerek web sayfasını oluşturur.

Web Siteleri ve Dinamik İçerik

- Web siteleri, statik (önceden depolanmış) ve dinamik (her istekte oluşturulan) içerik barındırır.
- Dinamik içerik, veritabanları gibi diğer sistemlerden gelen bilgileri içerebilir.

Geliştirme Alanları

- Ön uç (front-end) geliştirme, kullanıcı arayüzü ve etkileşimlerle ilgilenirken; arka uç (back-end) geliştirme, sunucu tarafındaki iş mantığını ve veri güvenliğini yönetir.
- Tam yığın (full-stack) geliştiriciler, her iki alanda da bilgi ve deneyime sahiptir.

Geliştirici Araçları

- Kod editörleri ve Entegre Geliştirme Ortamları (IDE'ler), kod yazma ve yönetme süreçlerini kolaylaştırır.
- Popüler IDE örnekleri arasında Visual Studio Code, Eclipse ve NetBeans bulunmaktadır.

Statik ve dinamik içerik arasındaki temel farklar şunlardır:

Statik İçerik

- **Tanım**: Statik içerik, sunucuda önceden depolanmış ve değişmeyen verilerdir. Kullanıcılar sayfayı her ziyaret ettiğinde aynı içeriği görürler.
- **Örnekler**: HTML dosyaları, resimler ve metin belgeleri.
- **Performans**: Genellikle daha hızlı yüklenir çünkü sunucu, içeriği her istekte yeniden oluşturmak zorunda değildir.
- **Kullanım Durumu**: Basit web siteleri veya portföyler gibi değişmeyen bilgilerin sunulması için idealdir.

Dinamik İçerik

- **Tanım**: Dinamik içerik, kullanıcı taleplerine veya etkileşimlerine bağlı olarak değişen verilerdir. Her kullanıcı veya her ziyaret için farklı içerikler sunulabilir.
- **Örnekler**: Kullanıcı profilleri, veritabanı sorguları ile oluşturulan sayfalar ve sosyal medya akışları.

- **Performans**: Genellikle daha yavaş yüklenir çünkü sunucu, her istekte içeriği oluşturmak için veritabanı veya diğer kaynaklarla etkileşimde bulunur.
- **Kullanım Durumu**: E-ticaret siteleri, sosyal medya platformları ve kullanıcı etkileşimi gerektiren uygulamalar için uygundur.

Bu farklar, web geliştirme sürecinde hangi tür içeriğin kullanılacağına karar vermede önemli bir rol oynar.

Web Sitesinin Yapısı

- HTML, web sitelerinin fiziksel yapısını oluşturmak için kullanılır; metin, bağlantılar, resimler ve butonlar gibi öğeleri içerir.
- CSS, web sitelerinin stilini ve görünümünü düzenler; renkler, yazı tipleri ve düzen gibi özellikleri yönetir.

JavaScript ve Diğer Diller

- JavaScript, web sitelerine etkileşim eklemek için kullanılır; örneğin, bir giriş butonunun işlevselliğini sağlamak için HTML ve CSS ile birlikte çalışır.
- SASS ve LESS gibi yeni diller, CSS'in işlevselliğini artırarak daha hızlı ve düzenli stil sayfaları oluşturmayı sağlar.

Responsive ve Adaptive Tasarım

- Responsive tasarım, web sitelerinin farklı cihazlara otomatik olarak uyum sağlamasını sağlar.
- Adaptive tasarım, belirli ekran boyutlarına göre optimize edilmiş web sitesi sürümleri sunar.

React

- React, Facebook tarafından geliştirilmiş bir JavaScript kütüphanesidir; web sayfalarında bileşenler oluşturmak ve render etmek için kullanılır.
- Tam bir araç seti sunmaz; yönlendirme gibi ek işlevsellikler için üçüncü taraf araçların eklenmesi gerekmektedir.

Angular

- Angular, Google tarafından sürdürülen açık kaynaklı bir framework'tür; HTML sayfalarını hızlı ve verimli bir şekilde render etmek için kullanılır.
- Yönlendirme ve form doğrulama gibi yerleşik araçlar sunar, bu da geliştiricilerin uygulama geliştirme sürecini kolaylaştırır.

Vue

- Vue, topluluk tarafından sürdürülen bir framework'tür; kullanıcı arayüzü ve görsel bileşenler üzerinde yoğunlaşır.
- Esnek, ölçeklenebilir ve diğer framework'lerle iyi bir şekilde entegre olabilme özelliğine sahiptir; hem bir kütüphane hem de bir framework olarak kullanılabilir.

Arka Uç Geliştirmenin Rolü

- Arka uç geliştiriciler, kullanıcıların istemciden yaptığı talepleri işlemek için gerekli kaynakları oluşturur ve yönetir.
- Kullanıcı bilgileri, ürün aramaları ve ödeme bilgileri gibi verilerin güvenli bir şekilde işlenmesi arka uç geliştiricilerin sorumluluğundadır.

Ön Uç ve Arka Uç İşbirliği

- Ön uç ve arka uç geliştiriciler, projelerin gereksinimlerini anlamak ve etkileşimlerini planlamak için yakın bir şekilde çalışmalıdır.
- Web uygulamaları ve bulut uygulamaları yaşam döngüsü boyunca işbirliği yaparak sorunları çözmek ve işlevsellik eklemek için birlikte çalışırlar.

Arka Uç Geliştirme Araçları ve Dilleri

- Arka uç geliştirme için JavaScript, Python gibi diller ve Node.js, Express, Django, Flask gibi çerçeveler yaygın olarak kullanılır.
- Veritabanlarıyla etkileşimde bulunmak için SQL bilgisi ve nesne ilişkisel haritalama (ORM) araçları da önemlidir.

Takım Çalışmasının Tanımı

- Takım, ortak bir hedefe ulaşmak için bir araya gelen farklı beceri ve deneyimlere sahip bireylerden oluşur.
- İşbirliği, yaratıcılığı artırır ve bireylerin yeteneklerini geliştirmelerine olanak tanır.

Başarılı Takım Çalışması İçin Gerekenler

- Güven ve saygı: Takım üyeleri arasında güven inşa edilmesi zaman alabilir, ancak eşit katkı sağlamak önemlidir.
- Hedeflerin belirlenmesi: Proje için ortak hedeflerin tanımlanması, takımın neye odaklanması gerektiğini belirler.
- İletişim: Etkili iletişim yöntemlerinin seçilmesi, tüm takımın bilgiye erişimini ve yanıt vermesini sağlar.

Yazılım Mühendisliğinde Takım Çalışması

- Proje başlangıcında kick-off toplantıları düzenlenir ve görevler atanır.
- Proje süresince düzenli toplantılar yapılır ve tasarım ile kod incelemeleri gerçekleştirilir.
- Proje tamamlandığında, retrospektif toplantılar yapılarak süreç değerlendirilir.

Takım Çalışmasının Faydaları

- Yaratıcılığı teşvik eder ve bireylerin güçlü yönlerinden faydalanmayı sağlar.
- Daha kaliteli kod yazımını ve daha az hata ile daha sürdürülebilir yazılımlar üretmeyi destekler.

- Takım içinde stresin azalmasına ve sorunların daha hızlı çözülmesine yardımcı olur.

Agile metodolojileri takip eden organizasyonlarda, küçük takımlara "squad" denir ve genellikle 10 kişiye kadar geliştiriciden oluşur.

Pair programming, iki geliştiricinin aynı bilgisayar başında yan yana çalıştığı bir işbirliği tekniğidir. Bu yöntem, sürekli planlama ve tartışma ile daha iyi bir son ürün elde etmeyi amaçlar.

Pair Programming Nedir?

- İki geliştirici, fiziksel olarak aynı bilgisayarda veya sanal ortamda çalışabilir.
- Agile geliştirme yönteminin bir parçasıdır ve sürekli iletişim gerektirir.

Farklı Pair Programming Stilleri

- **Driver/Navigator Stili**: Bir geliştirici kodu yazarken, diğeri kodu gözden geçirir ve yönlendirme yapar. Rollerin düzenli olarak değiştirilmesi önemlidir.
- **Ping-Pong Stili**: Her görevde bir geliştirici bir test yazar, diğeri bu testi geçecek kodu yazar. Rollerin sürekli değiştirilmesi sağlanır.
- **Strong Style**: Daha deneyimli bir geliştirici, daha az deneyimli olanın öğrenmesine yardımcı olur. Fikirler, uygulama tamamlanana kadar tartışılmaz.

Pair Programming'in Avantajları

- Bilgi ve beceri paylaşımını artırır.
- İki göz, daha az hata ve daha iyi kod kalitesi sağlar.
- İletişim ve problem çözme becerilerini geliştirir.

Karşılaşılan Zorluklar

- Uzun süreli odaklanma gerektirir, bu da yorucu olabilir.
- Bireysel kişiliklerin uyumsuzluğu sorun yaratabilir.
- Çalışma ortamında gürültüye neden olabilir.

Versiyon Kontrolü

- Versiyon kontrol sistemleri, kod değişikliklerini takip eder ve birden fazla geliştirici aynı projede çalışırken çakışmaları çözer.
- Git ve GitHub, kaynak kodu depolama ve yönetimi için popüler araçlardır; dosyaları depolayarak değişiklikleri izler ve kodu farklı dallara ayırarak daha odaklı geliştirme sağlar.

Kütüphaneler

- Kütüphaneler, belirli işlevleri veya özellik setlerini eklemek için kullanılabilecek kod koleksiyonlarıdır; örneğin, jQuery DOM manipülasyonunu basitleştirir.

- Kütüphaneler, kodun yeniden kullanılmasını sağlayarak uygulama geliştirmeyi hızlandırır ve geliştiriciye kontrol imkanı sunar.

Çerçeveler

- Çerçeveler, uygulamaları inşa etmek ve dağıtmak için standart bir yol sağlar; geliştiricinin kodunu ekleyebileceği bir iskelet gibi düşünülebilir.
- Çerçeveler, belirli bir yapı ve akış sunar, bu da geliştiricinin kontrolünü azaltabilir, ancak standartlaşmayı ve verimli kod yazımını teşvik eder. Örnekler arasında AngularJS ve Django bulunmaktadır.

Versiyon kontrol sisteminin temel işlevleri şunlardır:

- **Değişiklik Takibi**: Projeye yapılan her değişikliği kaydeder, böylece hangi değişikliklerin ne zaman ve kim tarafından yapıldığını görebilirsiniz.
- **Versiyon Yönetimi**: Farklı versiyonlar arasında geçiş yapma imkanı sunar. Geliştiriciler, önceki sürümlere geri dönebilir veya belirli bir versiyonu inceleyebilir.
- **Çatışma Çözümü**: Birden fazla geliştirici aynı dosya üzerinde çalıştığında ortaya çıkan çakışmaları yönetir ve çözer.
- **Dallandırma (Branching)**: Geliştiricilerin ana koddan bağımsız olarak yeni özellikler veya düzeltmeler üzerinde çalışmasına olanak tanır. Bu, farklı geliştirme yollarını paralel olarak sürdürmeyi sağlar.
- **Birleştirme (Merging)**: Dallarda yapılan değişikliklerin ana koda entegre edilmesini sağlar. Bu, farklı geliştiricilerin çalışmalarını bir araya getirmeyi kolaylaştırır.
- **Yedekleme**: Proje dosyalarının yedeklenmesini sağlar, böylece veri kaybı durumunda geri yükleme yapılabilir.

Bu işlevler, yazılım geliştirme sürecini daha düzenli ve verimli hale getirir.

CI/CD ve Build Tools

- CI/CD, sürekli entegrasyon ve sürekli teslimat veya dağıtım uygulamalarını ifade eder; bu, geliştiricilerin sık değişiklikleri güvenilir bir şekilde sunmalarını sağlar.
- Build araçları, kaynak kodunu kurulum için gerekli ikili dosyalara dönüştürür ve bağımlılıkları yönetir.

Paketler ve Paket Yöneticileri

- Paketler, uygulama dosyalarını, kurulum talimatlarını ve gerekli meta verileri içeren arşiv dosyalarıdır.
- Paket yöneticileri, yazılım paketlerini bulma, yükleme, güncelleme ve kaldırma işlemlerini otomatikleştirir.

Kullanım Örnekleri

- Popüler build araçları arasında Webpack, Babel ve Web Assembly yer alır.

- Farklı platformlar için yaygın paket yöneticileri arasında Linux için DPKG, Windows için Chocolatey ve MacOS için Homebrew bulunmaktadır.

- **Sürekli Entegrasyon (CI)**:
    - **Kod Deposu**: Geliştiricilerin kodlarını paylaştığı merkezi bir alan.
    - **Otomatik Testler**: Her kod değişikliği sonrası otomatik olarak çalışan testler, hataları erken tespit eder.
    - **Build Süreci**: Kodun derlenmesi ve çalışabilir bir uygulama haline getirilmesi.
- **Sürekli Teslimat (CD)**:
    - **Otomatik Dağıtım**: Başarılı bir build sonrası kodun test veya staging ortamına otomatik olarak dağıtılması.
    - **Kullanıcı Geri Bildirimi**: Kullanıcıların yeni özellikler hakkında geri bildirim vermesi için test ortamında uygulamanın sunulması.
- **Sürekli Dağıtım (CD)**:
    - **Üretim Ortamına Dağıtım**: Her başarılı build sonrası kodun otomatik olarak üretim ortamına dağıtılması.
    - **Gözlem ve İzleme**: Uygulamanın performansını ve hatalarını izlemek için araçlar kullanmak.

Yazılım Yığınları

- Yazılım yığını, uygulama geliştirmek için kullanılan yazılım ve programlama dillerinin bir kombinasyonudur.
- Yığınlar, kullanıcıya hizmet eden üst katmanlar ile donanım ile etkileşime giren alt katmanlar arasında hiyerarşik bir yapı oluşturur.

Yığın Türleri

- Yaygın yazılım yığınları arasında Python-Django, Ruby on Rails ve ASP.NET bulunmaktadır.
- LAMP, MEAN, MEVN ve MERN gibi diğer yığınlar, belirli teknolojilere dayalı olarak farklı uygulama geliştirme ihtiyaçlarını karşılar.

Avantajlar ve Dezavantajlar

- MEAN yığını, tüm bileşenlerin JavaScript kullanması nedeniyle geliştiricilere tek bir dil öğrenme avantajı sunar, ancak büyük ölçekli uygulamalar için uygun olmayabilir.
- LAMP yığını, ilişkisel verilerle iyi çalışırken, esneklik açısından MEAN ve MEVN yığınlarına göre daha sınırlıdır.

1. **MEAN Yığını**

MEAN yığını, JavaScript tabanlı bir yığıntır ve aşağıdaki bileşenlerden oluşur:

- **MongoDB**: NoSQL veritabanı olarak kullanılır. Verileri esnek bir şekilde depolar ve büyük ölçekli uygulamalar için uygundur.
- **Express.js**: Node.js üzerinde çalışan bir web uygulama çerçevesidir. Sunucu tarafında HTTP isteklerini yönetir ve API'ler oluşturmak için kullanılır.
- **Angular.js**: Kullanıcı arayüzü geliştirmek için kullanılan bir JavaScript çerçevesidir. Tek sayfa uygulamaları (SPA) oluşturmak için idealdir.
- **Node.js**: Sunucu tarafında JavaScript çalıştırmak için kullanılan bir platformdur. Geliştiricilerin JavaScript ile sunucu tarafı kodu yazmalarını sağlar.

Avantajları: Tüm bileşenlerin JavaScript kullanması, geliştiricilerin tek bir dilde uzmanlaşmasını sağlar. Ayrıca, açık kaynak olması nedeniyle maliyet avantajı sunar.

2. **LAMP Yığını**

LAMP yığını, web uygulamaları geliştirmek için kullanılan klasik bir yığıntır ve şu bileşenlerden oluşur:

- **Linux**: İşletim sistemi olarak kullanılır. Güvenilir ve açık kaynak bir platformdur.
- **Apache**: HTTP sunucusu olarak görev yapar. Web sayfalarını kullanıcıya sunmak için kullanılır.
- **MySQL**: İlişkisel veritabanı yönetim sistemi. Verileri yapılandırılmış bir şekilde depolar ve sorgular.
- **PHP**: Sunucu tarafında çalışan bir programlama dilidir. Dinamik web sayfaları oluşturmak için kullanılır.

Avantajları: LAMP, açık kaynak bileşenleri sayesinde geniş bir topluluk desteğine sahiptir ve esnek bir yapı sunar. Ayrıca, ilişkisel verilerle iyi çalışır.

3. **MERN Yığını**

MERN yığını, MEAN yığınına benzer, ancak Angular yerine React kullanır. Bileşenleri şunlardır:

- **MongoDB**: Veritabanı olarak kullanılır ve esnek veri yapıları sunar.
- **Express.js**: Sunucu tarafında HTTP isteklerini yönetir ve API'ler oluşturur.
- **React**: Kullanıcı arayüzü geliştirmek için kullanılan bir JavaScript kütüphanesidir. Bileşen tabanlı mimarisi sayesinde kullanıcı arayüzlerini hızlı bir şekilde oluşturmayı sağlar.
- **Node.js**: Sunucu tarafında JavaScript çalıştırmak için kullanılır.

Avantajları: React, yüksek performans ve esneklik sunar. Bileşen tabanlı yapısı sayesinde yeniden kullanılabilir bileşenler oluşturmak kolaydır.

Bu yığınlar, farklı uygulama geliştirme ihtiyaçlarına göre çeşitli avantajlar sunar.