React Native Projesinin Temel Yapısı

React Native projesinin temel yapısı, modülerliği ve ölçeklenebilirliği ön planda tutarak düzenlenmiştir. Bu yapı sayesinde, uygulamanın çeşitli bölümleri arasında tutarlılık sağlanabilir ve geliştiriciler arasında işbirliği kolaylaştırılabilir. Bu yapıyı anladığınızda, proje yönetimi, hata ayıklama ve kodun sürdürülebilirliğini sağlama konularında daha etkin hale gelebilirsiniz.

1. Root (Ana) Dizin

- node\_modules/: Proje tarafından kullanılan tüm bağımlılıkların bulunduğu klasördür. package.json dosyasında belirtilen bağımlılıklar, bu klasör içinde yer alır.

- ios/ ve android/: React Native, iOS ve Android platformlarını desteklediği için bu klasörler, her iki platforma özgü yapılandırma ve kaynak dosyalarını içerir. Örneğin, iOS için `Objective-C` veya `Swift` ile yazılmış kodlar `ios/` klasöründe bulunurken, Android için `Java` veya `Kotlin` ile yazılmış kodlar `android/` klasöründe bulunur.

- App.js veya index.js: Uygulamanın giriş noktasıdır. Bu dosya, uygulamanın ilk olarak yükleneceği bileşeni belirler ve genellikle kök bileşen burada tanımlanır. Uygulamanın genel yapısını buradan başlatabilirsiniz.

- package.json: Projenin bağımlılıklarını ve yapılandırmalarını listeleyen dosyadır. Bu dosyada proje adı, sürümü, bağımlılıkları ve script'ler gibi bilgiler yer alır. Yeni bağımlılıklar eklendiğinde veya mevcut bağımlılıklar güncellendiğinde bu dosya güncellenir.

- babel.config.js: Babel, modern JavaScript kodunu eski tarayıcılar veya JavaScript motorlarıyla uyumlu hale getiren bir derleyicidir. Bu dosya, projenin Babel ayarlarını içerir.

- metro.config.js: Metro, React Native için varsayılan modül paketleyicidir. Bu dosya, Metro'nun nasıl çalışacağını ve hangi modülleri paketleyeceğini tanımlar.

2. src/ Klasörü

Proje yapınızı daha düzenli ve modüler hale getirmek için tüm kaynak kodlarını src/ klasöründe toplamak yaygın bir yaklaşımdır. Bu klasör, uygulamanın ana bileşenlerini, stillerini, hizmetlerini, yapılandırmalarını ve diğer önemli unsurlarını içerebilir. src/ klasörünün yapısı genellikle aşağıdaki gibidir:

- components/: Tekrar kullanılabilir React bileşenleri bu klasörde yer alır. Örneğin, Button, Header, Footer gibi genel amaçlı bileşenler burada bulunur.

- screens/: Her bir uygulama ekranı için ayrı bir dosya veya klasör bu dizinde yer alır. Örneğin, HomeScreen, LoginScreen, ProfileScreen gibi ekranlar burada bulunur.

- navigation/: Uygulamanın navigasyon yapısı burada tanımlanır. React Navigation gibi bir kütüphane kullanıyorsanız, buradaki dosyalarda navigasyon yapılandırmalarını bulabilirsiniz.

- assets/: Uygulamada kullanılan statik dosyalar, örneğin görüntüler, ikonlar, fontlar ve diğer medya dosyaları burada saklanır.

- styles/: Uygulamanın genel stil dosyaları burada bulunur. Bu dosyalar, uygulamanın farklı bölümleri arasında tutarlılığı sağlamak için ortak stilleri barındırabilir.

- utils/ veya helpers/: Uygulamanın çeşitli yerlerinde kullanılabilecek yardımcı fonksiyonlar ve yardımcı modüller bu klasörde yer alır.

- context/: Uygulamanın global state yönetimi için Context API veya başka bir state yönetim kütüphanesi (örn. Redux) kullanılıyorsa, bu yapılandırmalar burada bulunur.

- services/: API çağrıları, veri çekme işlemleri gibi hizmetler bu klasörde yer alır. Burada, uygulamanızın backend ile iletişimini yöneten tüm fonksiyonları bulabilirsiniz.

3. Diğer Yapılandırma Dosyaları

- .gitignore: Bu dosya, git sürüm kontrol sistemi tarafından izlenmemesi gereken dosya ve klasörleri tanımlar. Genellikle node\_modules/, ios/build, android/build, ve bazı platforma özgü yapı dosyaları bu dosya tarafından göz ardı edilir.

- tsconfig.json: Eğer TypeScript kullanıyorsanız, bu dosya TypeScript yapılandırmalarını içerir. TypeScript, JavaScript'in statik tip denetimine sahip bir süpersetidir.

4. Expo Projeleri İçin Özel Dosyalar

Eğer Expo kullanıyorsanız, Expo'ya özel bazı ek dosyalar da proje yapınıza dahil edilir:

- app.json: Expo projeleri için temel yapılandırma dosyasıdır. Bu dosya, uygulamanın adı, ikonu, başlatma ekranı gibi genel bilgileri içerir.

- app.config.js: app.json dosyasının dinamik bir versiyonudur ve JavaScript kullanarak yapılandırma yapmanıza olanak tanır.

5. Uygulama Akışı

React Native uygulamaları, React bileşenleri üzerine kurulu bir yapıdadır. Her bileşen, uygulamanın belirli bir bölümünü temsil eder. Uygulama, index.js dosyasından başlar ve buradan kök bileşeni (App.js) yükler. Uygulamanın geri kalan kısmı bu kök bileşenden dallanır.

Uygulamanın tüm ekranları, screens/ klasöründe tanımlanır ve genellikle bir navigasyon yapısı (navigation/) aracılığıyla bir araya getirilir. Kullanıcı bir ekranla etkileşime girdiğinde, ilgili bileşenler (components/) çağrılır ve bu bileşenler gerekli veri işlemlerini yaparak (genellikle services/ klasöründen API çağrıları) kullanıcıya gerekli bilgileri sunar.

Tekrar kullanılabilir component nasıl dizayn edilir?

Tekrar kullanılabilir bir component, farklı yerlerde aynı işlevi yerine getiren ve yeniden kod yazma gereksinimini ortadan kaldıran bir yapı olarak tasarlanır. Bu, kodun daha temiz, sürdürülebilir ve bakımının kolay olmasını sağlar. React Native'de tekrar kullanılabilir bir komponent tasarlamak için şu adımları izleyebilirsiniz:

1. Komponentin Amacını Belirleme

İşlevsellik: Komponentin ne yapacağını ve hangi işlevleri yerine getireceğini belirleyin. Örneğin, bir Buttonkomponenti, bir butonun nasıl görüneceğini ve nasıl tepki vereceğini belirler.

Yeniden Kullanım Senaryoları: Bu komponentin nerelerde ve nasıl kullanılacağını düşünün. Bu, komponentin hangi özelliklerinin (props) özelleştirilebilir olması gerektiğini anlamanıza yardımcı olur.

2. Props ile Özelleştirme

Özelliklerin Tanımlanması: Komponentin farklı senaryolarda nasıl davranacağını kontrol etmek için props tanımlayın. Örneğin, Button komponenti için title, onPress, color, disabled gibi props tanımlanabilir.

Varsayılan Değerler (defaultProps): Eğer belirli bir prop verilmezse komponentin nasıl davranacağını belirlemek için varsayılan değerler kullanın. Bu, komponentin daha esnek ve kullanımının daha kolay olmasını sağlar.

Prop Türlerinin Belirlenmesi (propTypes): Props için beklenen veri türlerini belirleyin. Bu, komponentin doğru bir şekilde kullanılmasını sağlar ve hata ayıklamayı kolaylaştırır.

3. Stil ve Düzenleme

Modüler Stil: Komponentin stilini esnek ve modüler yapın. Bu, komponentin farklı temalarda veya düzenlerde kullanılabilmesini sağlar. Stil bilgilerini, props üzerinden alarak dinamik hale getirebilirsiniz.

Stil Bileşenleri: StyleSheet kullanarak stil bileşenleri tanımlayın ve komponent içinde bu stilleri kullanarak düzenli bir görünüm sağlayın.