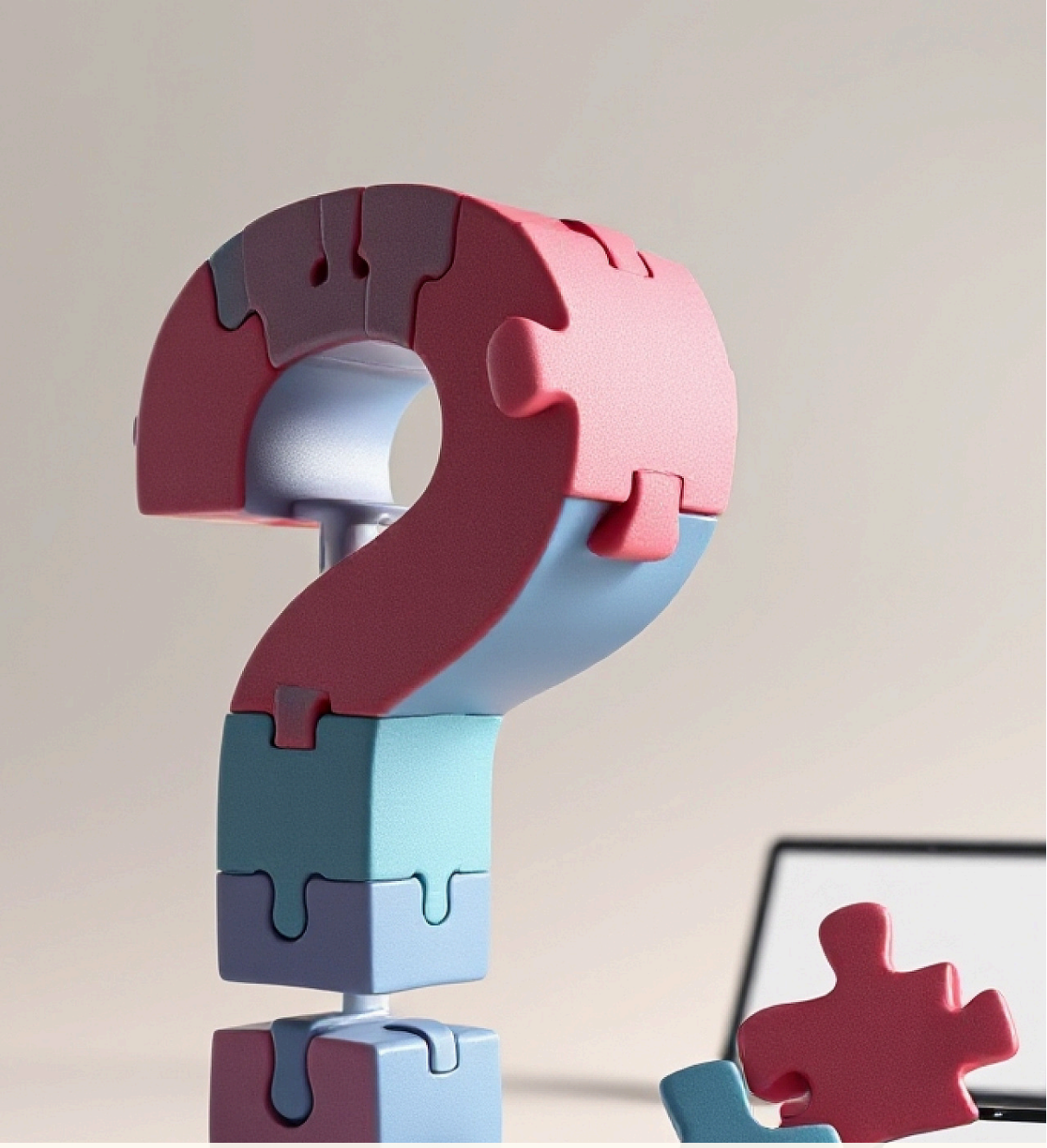
Kodda dal oluşturmanın amacı nedir?



www.yzokul.com.tr

Kodda dal oluşturmanın amacı nedir?



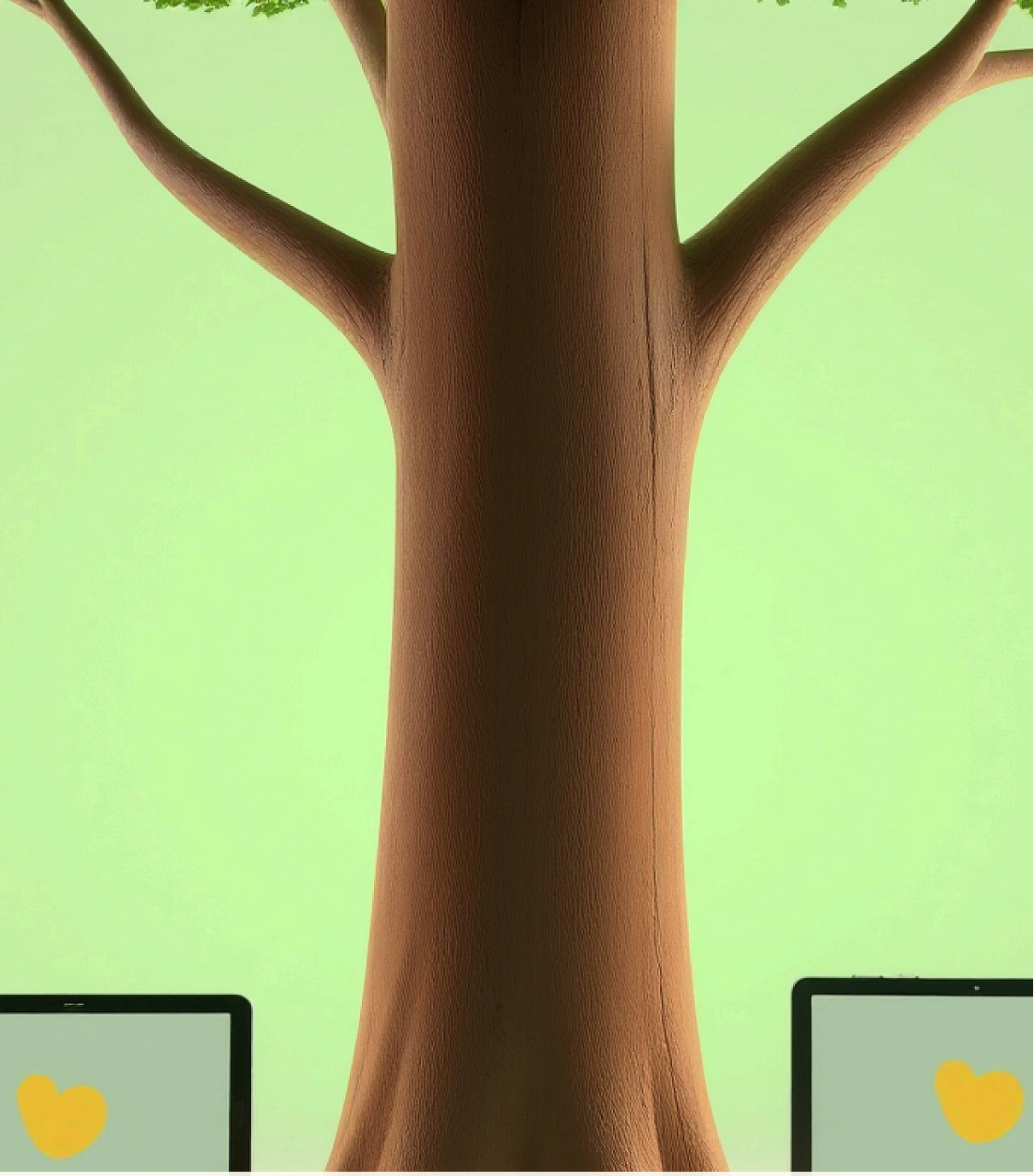
Muhtemelen şu soruyu görmüşsündür: kodda dal oluşturmanın amacı nedir? Kısa yanıt, bir dalın, ana sürüm kullanılabilir kalırken projeyi ayrı bir geliştirme hattı boyunca değiştirmene olanak tanımasıdır. Bu, zaten çalışan şeyleri bozmadan yeni şeyler deneyebileceğin anlamına gelir. Dallar, dosyalardaki değişiklikleri zaman içinde izleyen bir sistem olan sürüm kontrolünün temel bir fikridir.



Bir projeyi kod dosyaları ve görseller gibi dosyalarla dolu bir klasör olarak düşün. Yeni özellikler veya düzeltmeler üzerinde çalışırken değiştireceğin şey bu klasördür. Bir dal, bu değişiklikleri ana projeden ayrı olarak düzenli bir şekilde yapmana olanak tanır. Örneğin, bir oyun projesinde grafikler, sesler ve oyun mantığı için dosyalar olabilir.



Bir proje, değişikliklerinin zaman içindeki geçmişini tutar. Bu geçmiş, neyin, ne zaman ve kim tarafından değiştirildiğini kaydeder. Proje üzerinde çalışan herkes aynı geçmişin üzerine inşa eder, böylece projenin nasıl evrildiğini görebilirsin. Dallar, bu geçmişi ayrı çalışma için bir başlangıç noktası olarak kullanır. Bu ortak kayıt, ekiplerin değişiklikleri takip etmeyi kaybetmeden işbirliği yapmasına yardımcı olur.



Bir dal, aynı proje içinde ayrı bir geliştirme hattıdır. Geçerli sürümden başlar ve ana hatta dokunmadan değişiklik yapmana olanak tanır. Yeni bir şey üzerinde çalışmak istediğinde bir dal oluşturabilirsin. Genellikle ana dal olarak adlandırılan ana hat, sen dalın üzerinde çalışırken değişmeden kalır. Bu ayrım, dal oluşturmanın temel amacıdır.



Bir dal üzerinde çalışırken ana sürüm eskisi gibi çalışmaya devam eder. İnsanlar onu hâlâ kullanabilir ve bitmemiş işin onu rahatsız etmez. Bu, acele etmeden denemeler yapabileceğin ve başkaları için hiçbir şeyi bozmayacağın anlamına gelir. Örneğin, yeni bir özellik ekliyorsan ana sürüm kararlı ve kullanılabilir kalır. Bu güvenlik, dal kullanmanın önemli bir nedenidir.



Birinin yeni bir menü yapmak istediği bir oyun projesi düşün. Menü çalışması için bir dal oluştururken oyunun ana sürümü oynanabilir kalır. Dal üzerinde, oyuncuların kullandığı oyunu etkilemeden menüyü tasarlayıp kodlayabilirler. Menü hazır olduğunda onu ana sürüme getirebilirler. Bu örnek, dalların paralel çalışmayı nasıl mümkün kıldığını gösterir.



Dal üzerinde şeyler deneyebilir, hatalar yapabilir ve dosyaları serbestçe değiştirebilirsin. Değişiklikleri geri getirmeye karar verene kadar bunların hiçbiri ana sürümü etkilemez. Ana projeyi bozma endişesi duymadan fikirlerini test edip geliştirebilirsin. Bu özgürlük denemeyi ve öğrenmeyi teşvik eder. Yeni başlayanlar için bu, kod pratiği yapmanın güvenli bir yoludur.



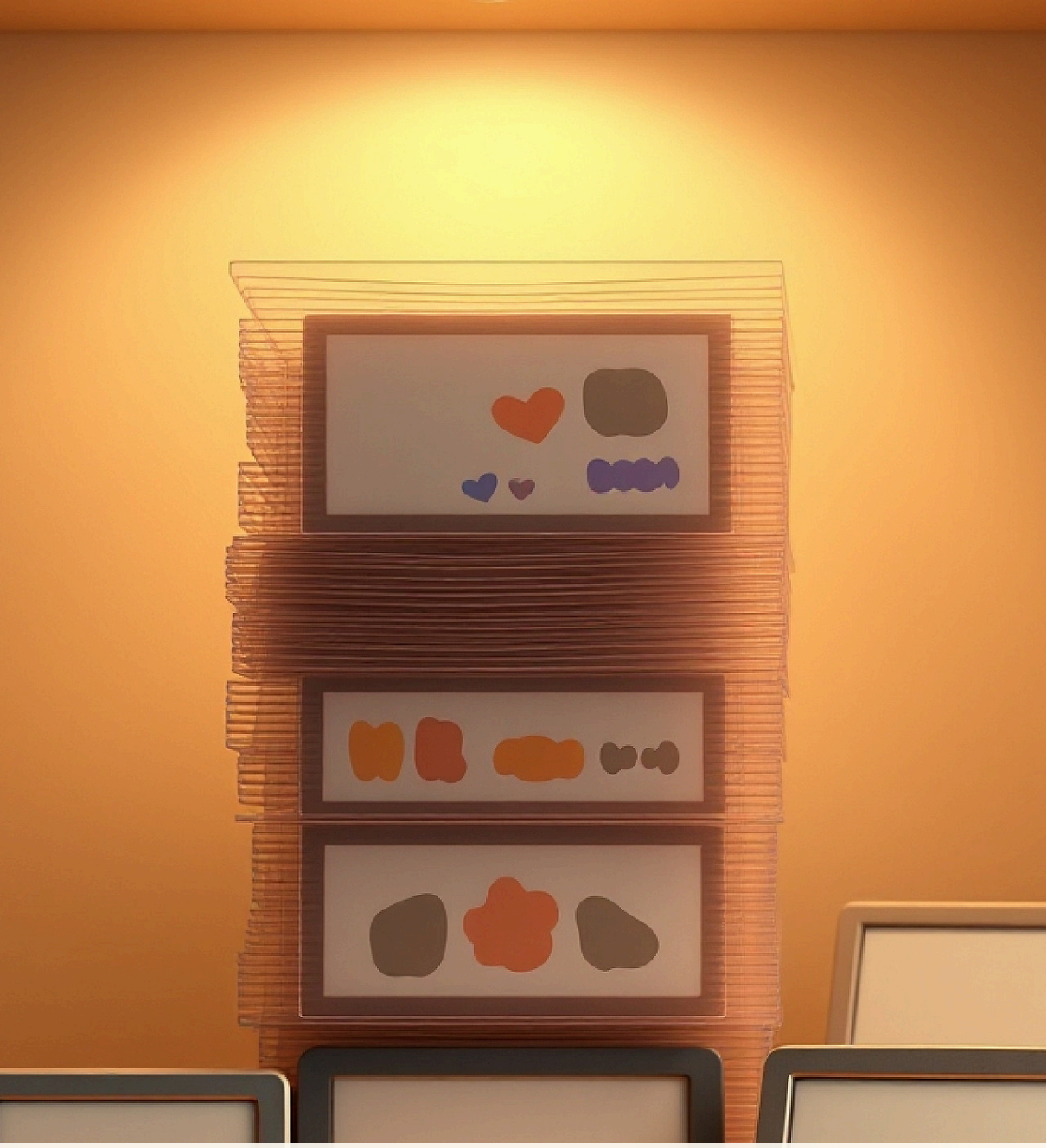
Bir dal tamamen bağımsız bir proje değildir. Aynı projeye aittir ve dosyalarını ve geçmişini paylaşır. Bir dal oluşturduğunda sıfırdan başlamazsın; zaten var olanın üzerine inşa edersin. Bu, kodu ve varlıkları çoğaltmadan yeniden kullanabileceğin anlamına gelir. Dallar hafiftir ve oluşturulması kolaydır, bu yüzden farklı görevler için çok sayıda dalın olabilir.



Yeni menü tamamlanıp çalıştığında dal işini yapmıştır. Artık bitmiş değişiklik ana hatta geri dönmeye hazırdır. Menüyü dal üzerinde test ettin ve çalıştığından eminsin. Sonraki adım, çalışmanı ana sürümle birleştirmektir. Birleştirme tam burada devreye girer.



Birleştirme, tamamlanan dal değişikliklerini uygun işlemlerle ana dalla bir araya getirme sürecidir. Birleştirmeden sonra ana sürüm artık yeni menüyü içerir. Birleştirme işlemi, dal üzerinde yaptığın değişiklikleri alıp ana dala uygular. Çalışman böylece ana projenin parçası olur. Birleştirme, sürüm kontrolünde temel bir işlemdir.



Dosyaları kopyalamak yerine neden birleştirme? Birleştirme, dalın değişiklik geçmişini dosyalarla birlikte getirir. Bu, projenin nasıl büyüdüğüne dair tek ve net bir kayıt tutar. Dosyaları yalnızca kopyalasaydın, kimin neyi neden değiştirdiğine dair geçmişi kaybederdin. Birleştirme, çalışmanın bağlamını korur ve projeyi zaman içinde anlamayı ve bakımını yapmayı kolaylaştırır.



Aynı dosyalar farklı dallarda değiştirildiyse birleştirme bunları her zaman otomatik olarak bir araya getiremez. Aracın hangi değişikliğin tutulacağına karar vermek için yardıma ihtiyacı vardır. Bu duruma çakışma denir. Çakışmalar, iki dal bir dosyanın aynı bölümünü farklı şekillerde değiştirdiğinde ortaya çıkar. Çakışmayı elle çözmen gerekecektir.



Çakışma, aynı dosyanın iki sürümünü karşılaştırıp doğru sonucu seçtiğin andır. Bundan sonra birleştirme tamamlanabilir. Değişiklikleri bir araya getirmek için dosyayı düzenlemen veya bir sürümü diğerine tercih etmen gerekebilir. Çakışmaları çözmek, dallarla işbirliği yapmanın normal bir parçasıdır. Son kodun doğru ve tutarlı olmasını sağlar.



Yaygın bir hata, büyük işler için ana sürümü doğrudan değiştirmektir; böylece ana sürüm bitmemişken bozulabilir. Bir dal, riskli işi ayrı tutarak bunu önler. Doğrudan ana dal üzerinde çalışırsan herkesi etkileyen hatalar getirebilirsin. Bir dal kullanarak ana projeyi aksatmadan denemeler yapıp test edebilirsin. Önemli değişiklikler için her zaman bir dal oluştur.



Bir başka hata, bitmiş işi sonsuza dek bir dalda bırakmaktır; böylece ana sürüm onu hiç almaz. Tamamlanan değişiklikler geri birleştirilmelidir. Birleştirmeyi unutursan çalışman ana projeye entegre edilmez ve başkaları bundan yararlanamaz. Birleştirmeyi iş akışının düzenli bir parçası haline getir. Bir özellik bitip test edildiğinde onu hemen birleştir.



Özetlemek gerekirse, bir dal ana sürüm kullanılabilir kalırken ayrı olarak geliştirme yapmana olanak tanır ve bitmiş iş geri birleştirilir; gerekirse çakışmalar çözülür. Unutma: ayrı çalışma için dal oluştur, iş bitince birleştir. Bu yaklaşım projeni kararlı, değişikliklerini düzenli tutar. Bu uygulamaları izleyerek etkili bir şekilde işbirliği yapabilir ve sağlıklı bir kod tabanını koruyabilirsin.