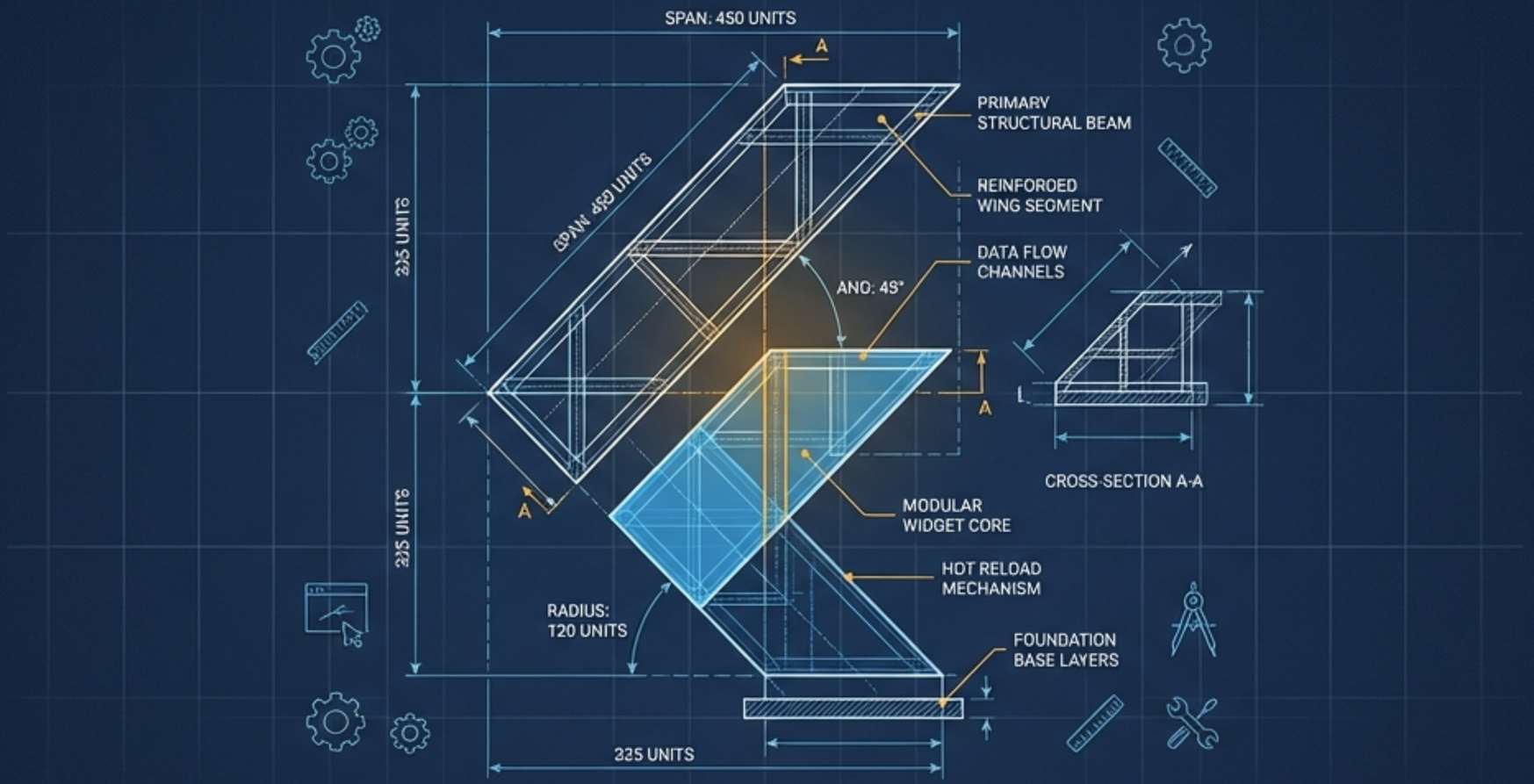


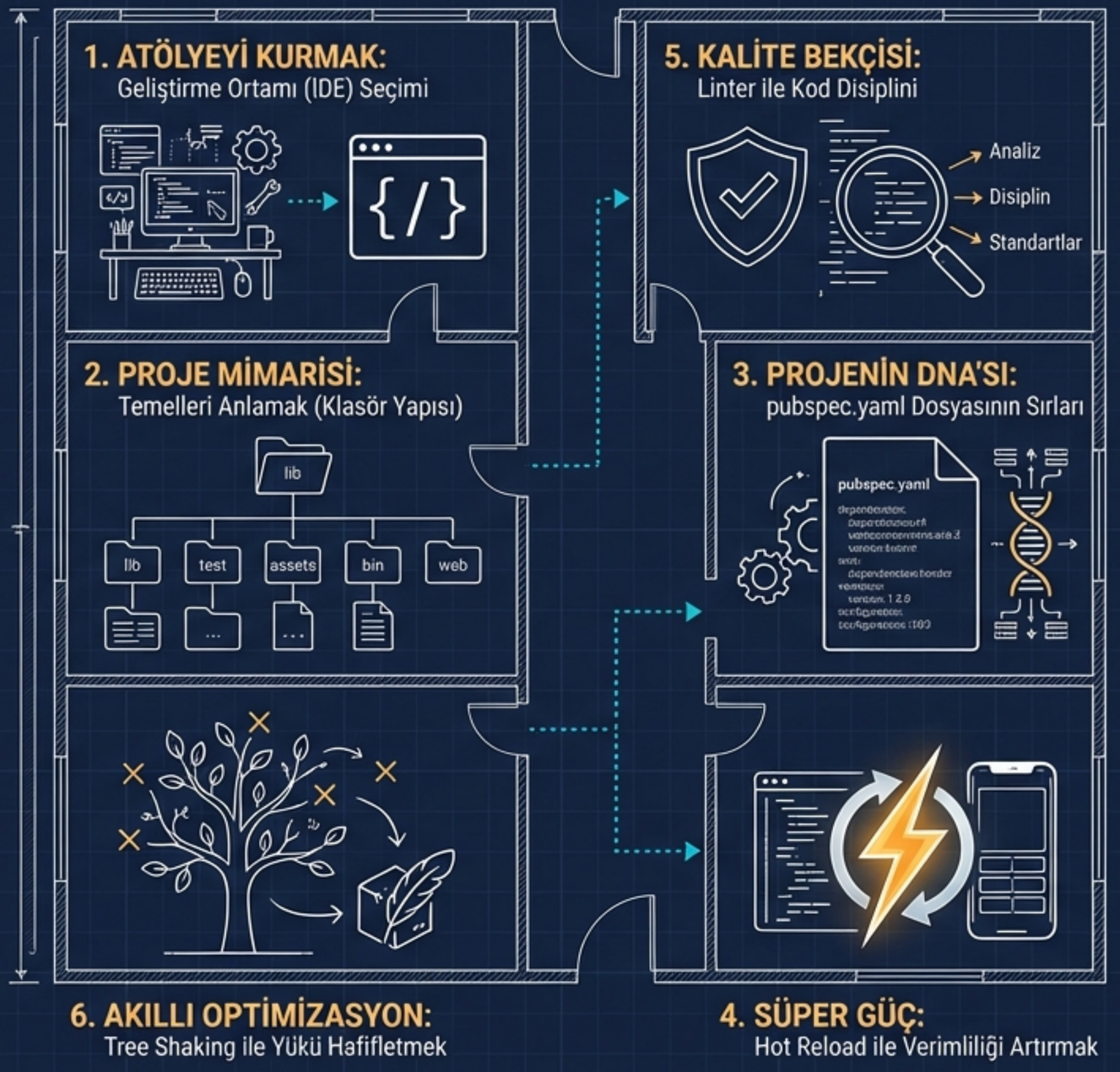
Flutter Projesinin Anatomisi

Sağlam Bir Başlangıç İçin Profesyonel Bir Kılavuz

Yapı • Araçlar • En İyi Pratikler

Keşif Turumuz: Başarının Planı

1. **Atölyeyi Kurmak:**
   Geliştirme Ortamı (IDE) Seçimi
2. **Proje Mimarisi:**
   Temelleri Anlamak (Klasör Yapısı)
3. **Projenin DNA'sı:**
   pubspec.yaml Dosyasının Sırları
4. **Süper Güç:**
   Hot Reload ile Verimliliği Artırmak
5. **Kalite Bekçisi:**
   Linter ile Kod Disiplini
6. **Akıllı Optimizasyon:**
   Tree Shaking ile Yükü Hafifletmek

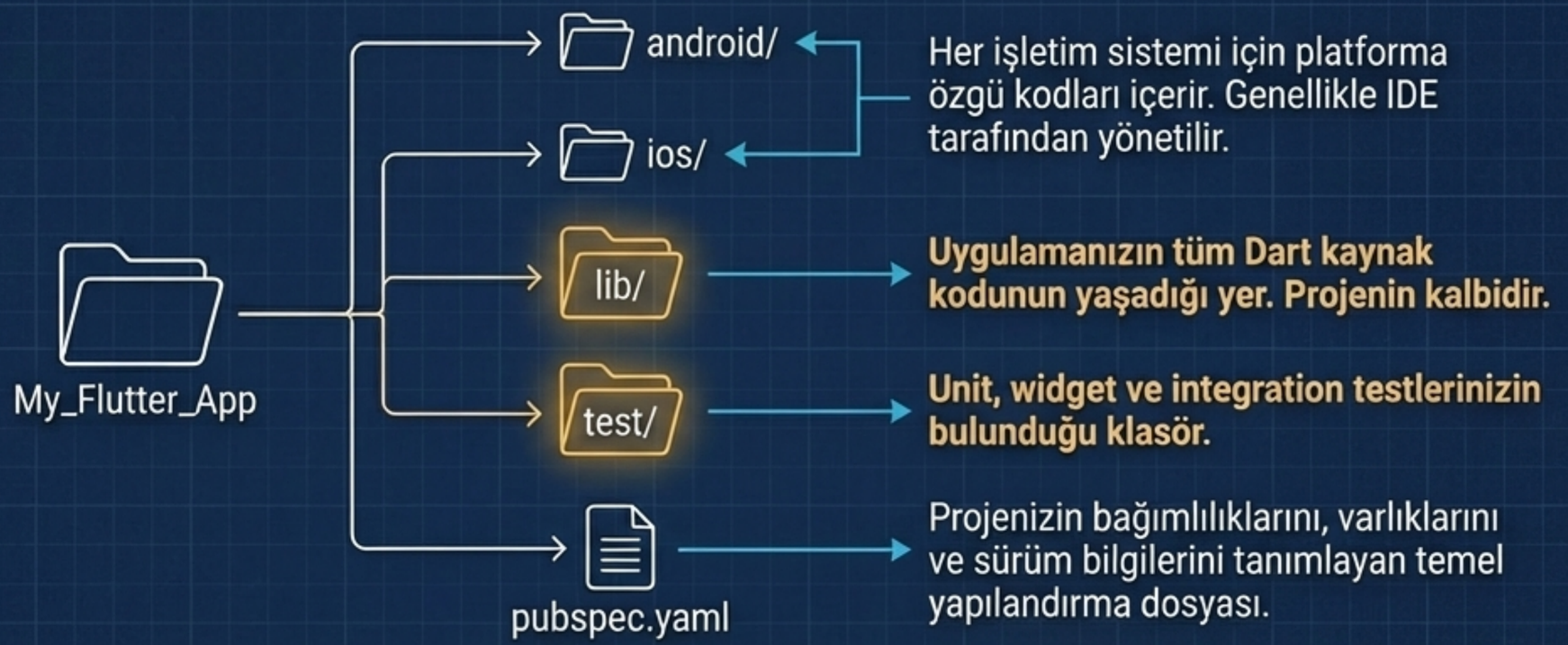


Atölyeyi Kurmak: Doğru Araçları Seçmek

- Flutter, Android Studio ve Visual Studio Code (VS Code) ile birinci sınıf bir geliştirme deneyimi sunar.
- Ancak, tercih ettiğiniz metin düzenleyici ve komut satırı araçlarıyla da geliştirme yapabilirsiniz.

Kilit Mesaj: Hangi IDE'yi seçerseniz seçin, projenizin temel mimarisi ve dosyaları aynı kalacaktır. Önemli olan bu mimariye hakim olmaktır.

Proje Mimarisi: Bir Flutter Projesinin Anatomisi

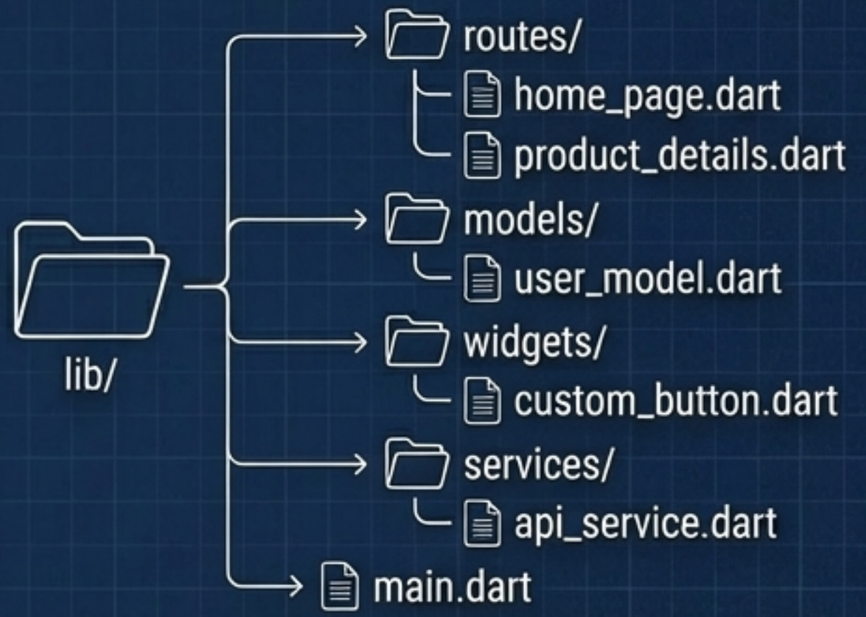


Zamanınızın %99'unu `lib/` ve `test/` klasörlerinde geçireceksiniz. Diğerlerinin ne işe yaradığını bilmek ise projenize hakim olmanızı sağlar.

Kodunuzun Kalbi: Ölçeklenebilir Bir `lib/` Mimarisi

Organize Olmamış Yapı

Önerilen Yapı



İyi bir başlangıç, gelecekteki baş ağrılarını önler. Organizasyon, projeniz büyüdükçe en değerli varlığınız olacaktır.

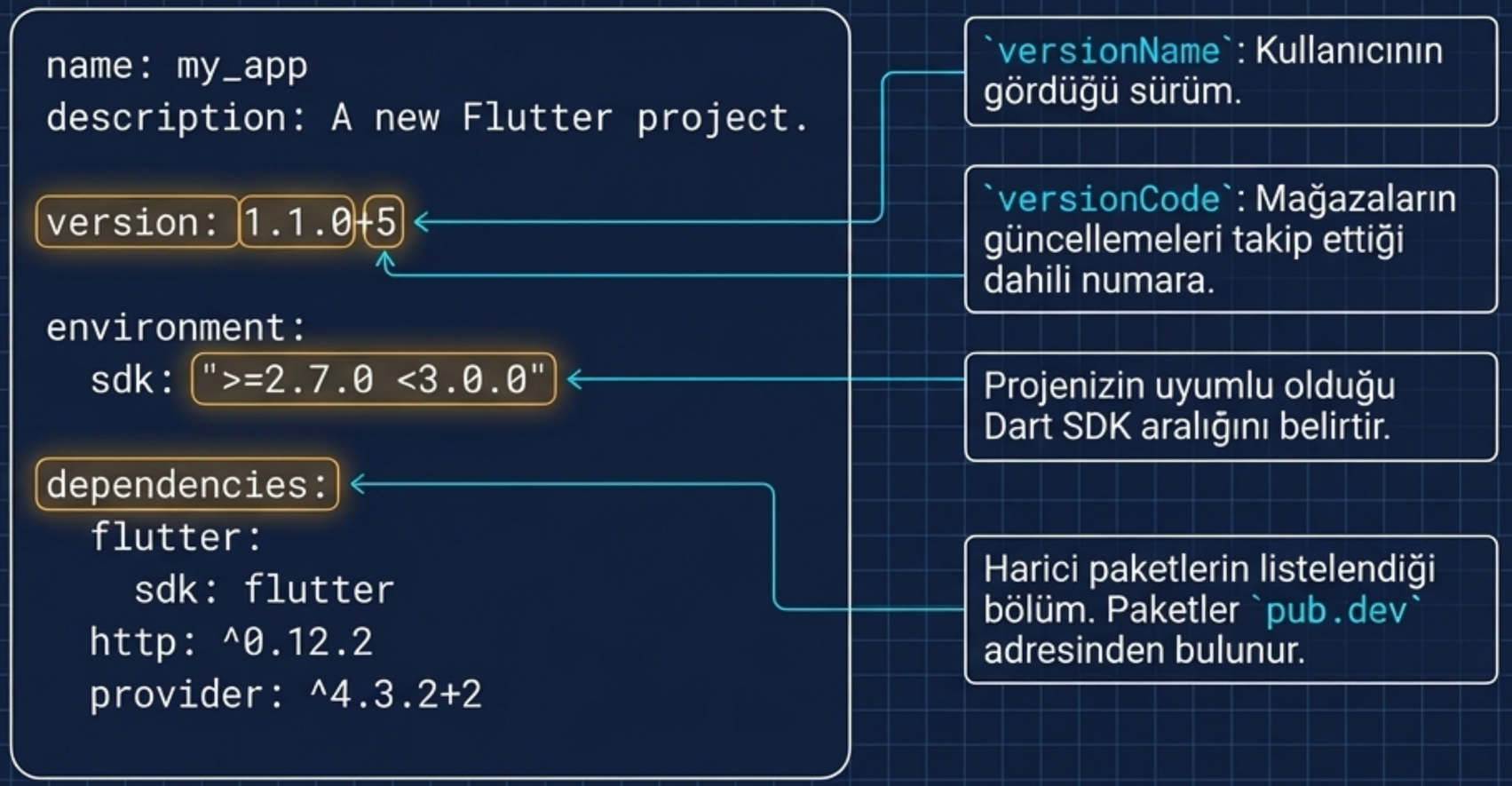
Projenin DNA’sı: `pubspec.yaml` Dosyasına Giriş

Bu dosya, projenizin kimlik kartı ve kontrol merkezidir. İnsan tarafından okunabilir bir format olan YAML kullanır ve girintilere (indentation) duyarlıdır.

Üç Temel Rolü:

1. **Bağımlılık Yönetimi:** Projenizin ihtiyaç duyduğu harici paketleri yönetir.
2. **Varlık (Asset) Tanımlaması:** Resimler, fontlar ve diğer statik dosyaları projenize dahil eder.
3. **Sürüm Kontrolü:** Uygulamanızın mağazalarda yayınlanacak sürüm numarasını belirler.

Derinlemesine Bakış: Bağımlılıklar ve Sürüm Yönetimi



Derinlemesine Bakış: Varlıklar (Assets) ve Fontlar

pubspec.yaml

```
flutter:
  uses-material-design: true

  assets:
    - images/
    - files/text/myFile.txt

  fonts:
    - family: Roboto
      fonts:
        - asset: fonts/Roboto-Regular.ttf
        - asset: fonts/Roboto-Italic.ttf
          style: italic
```

Uygulamadaki görsellerin bulunduğu klasör veya dosyalar. Tüm klasörü ekler.

Belirli veri dosyaları veya belgeler. Tekil dosya ekler.

Projede kullanılan özel yazı tiplerinin tanımları.

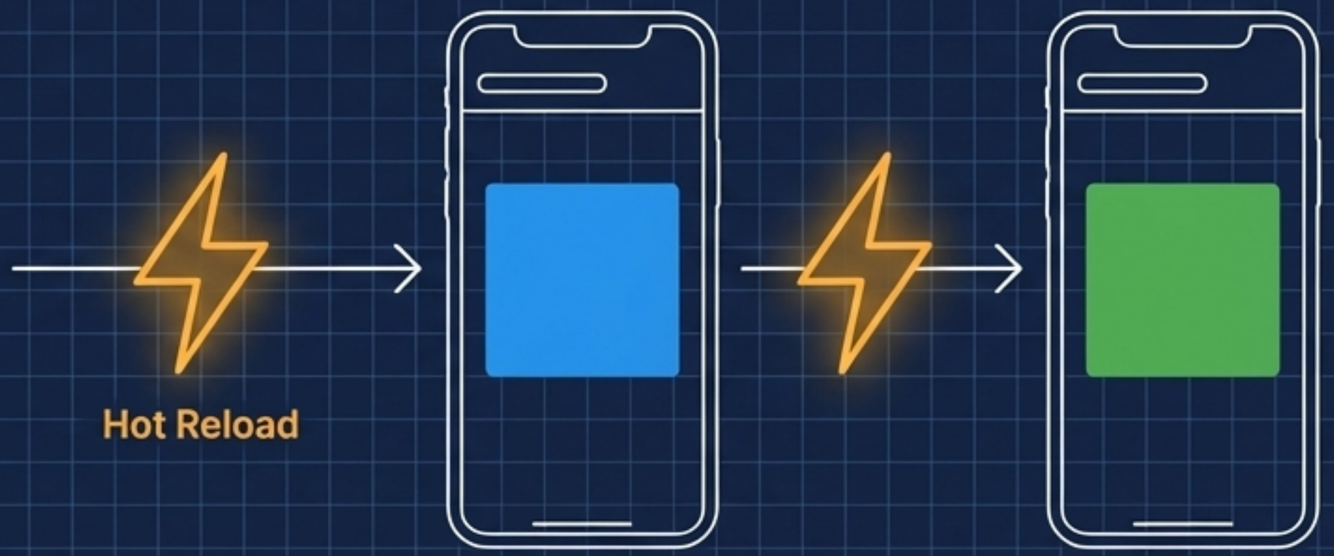
Pro-Tip

Geliştirme Kolaylığı, Yayın Güvenliği

`google_fonts` paketini geliştirme sırasında fontları internetten çekmek için kullanın. Ancak uygulamanızı yayınlamadan önce, kullandığınız fontları projenize bir `asset` olarak ekleyin. Bu, uygulamanızın internet bağlantısı olmadığında bile doğru fontlarla çalışmasını garanti eder ve daha hızlı yükleme sağlar.

Geliştiricinin Süper Gücü: Hot Reload

```
  class {
    app Epe: {
      color: Colors.blue,
    },
    usage view: {
      color: Colors.green,
    };
  }
}
```



Nedir?

Kodunuzda yaptığınız değişiklikleri, uygulamanın durumunu (state) kaybetmeden, bir saniyeden kısa sürede emülatör veya cihaz üzerinde görmenizi sağlayan özelliktir.

Nasıl Çalışır?

Debug modunda, Dart Sanal Makinesi ve JIT (Just-In-Time) derlemesi sayesinde kod değişiklikleri anında çalışan uygulamaya enjekte edilir.

Sınırlamalar:

`initState()`, `static` alanlar veya `class/enum` tanımlarında yapılan büyük değişiklikler için tam yeniden başlatma gerekebilir.

Kalite Bekçisi: Linter ile Kod Disiplini

- **Linter Nedir?** Kaynak kodunuzu analiz ederek stil hatalarını, olası bug'ları ve en iyi pratiklere uymayan yapıları tespit eden bir araçtır.
- **Neden Önemli?** Dart'ın varsayılan linter'ı oldukça esnektir. Daha katı kurallar belirlemek, kodunuzu daha güvenli, tutarlı ve okunabilir hale getirir.
- **Nasıl Yapılır?** Projenizin ana dizinine `analysis_options.yaml` adında bir dosya oluşturarak linter kurallarını özelleştirebilirsiniz.

Pro-Tip: `analysis_options.yaml` dosyası, Dart'ın en iyi pratiklerini hatırlamak zorunda kalmadan uygulamanızı sağlayan bir rehber görevi görür. IDE'niz bu dosyayı okur ve sizi anlık olarak uyarır.

Kuralları Belirlemek: `analysis_options.yaml` Yapılandırması

```yaml
analyzer:
  errors:
    dead_code: warning
    # Bir kuralın önemini artırmak için 'error' kullanın
    await_only_futures: error

linter:
  rules:
    # Aktif etmek istediğiniz kuralları listeleyin
    - await_only_futures
    - empty_constructor_bodies
    - prefer_const_constructors
    - slash_for_doc_comments
    - type_init_formals
```

Şiddet Seviyeleri

- **`error`**: Analizin başarısız olmasına neden olur.
- **`warning`**: Varsayılan seviye, analizi durdurmaz.
- **`info`**: Bilgilendirici mesaj.
- **`ignore`**: Kuralı tamamen yok sayar.

Kod standartlarınızı proje bazında tanımlayarak tüm ekibiniz için tutarlılık sağlayın.

Akıllı Optimizasyon: Sadece İhtiyacınız Olanı Paketleyin

Geliştirme Kodu (`kDebugMode` true)

```
if (kDebugMode) {
  print("Uygulama debug modda çalışıyor.");
} else {
  setupCrashAnalytics();
}
```

Yayın Kodu (Derlenmiş Hali)

```
setupCrashAnalytics();
```

Tree Shaking Nedir? Derleyicinin, uygulamanızın yayın (release) versiyonunu oluştururken kullanılmayan kodları ('ölü kod') otomatik olarak kaldırması işlemidir. Bu, nihai uygulama boyutunu küçültür.

Başarının Mimarisi: Profesyonel Projenin Özeti

Profesyonel Flutter Projesi

Organize Yapı

`lib/` ve `test/` klasörlerinin doğru yapılandırılması.

Merkezi Konfigürasyon

`pubspec.yaml` ile proje DNA'sının yönetimi.

Verimli Akış

`Hot Reload` ile geliştirme hızını katlama.

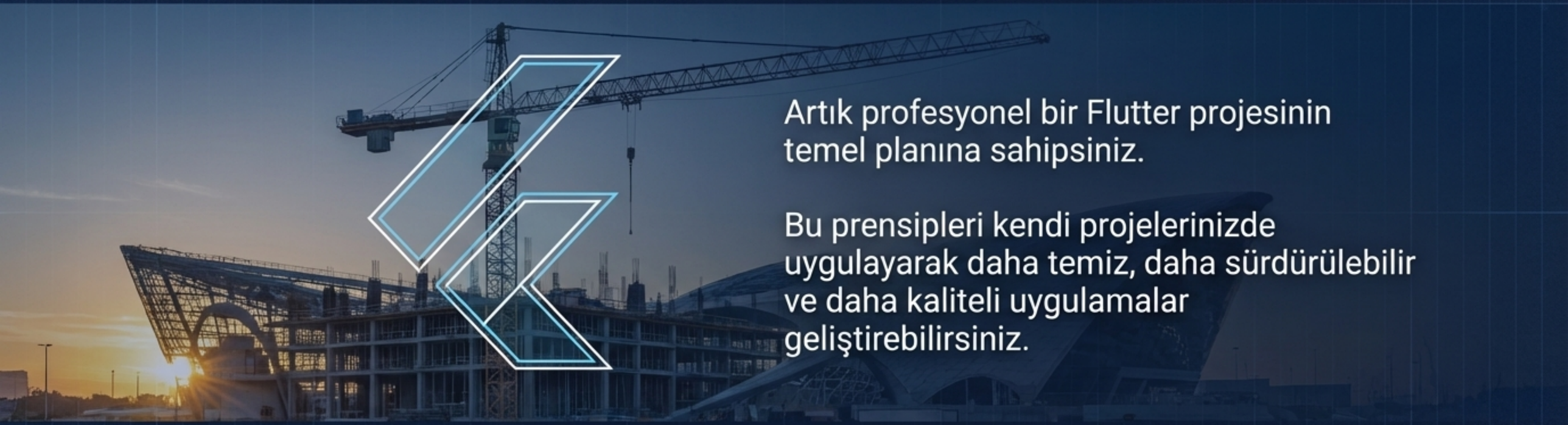
Dahili Kalite

`Linter` ile kod standartlarını otomatize etme.

Optimize Çıktı

`Tree Shaking` ile yalın ve performanslı uygulamalar.

İnşaata Başlayın



Artık profesyonel bir Flutter projesinin temel planına sahipsiniz.

Bu prensipleri kendi projelerinizde uygulayarak daha temiz, daha sürdürülebilir ve daha kaliteli uygulamalar geliştirebilirsiniz.

Daha fazla bilgi için resmi Flutter dokümantasyonunu keşfedin:

flutter.dev