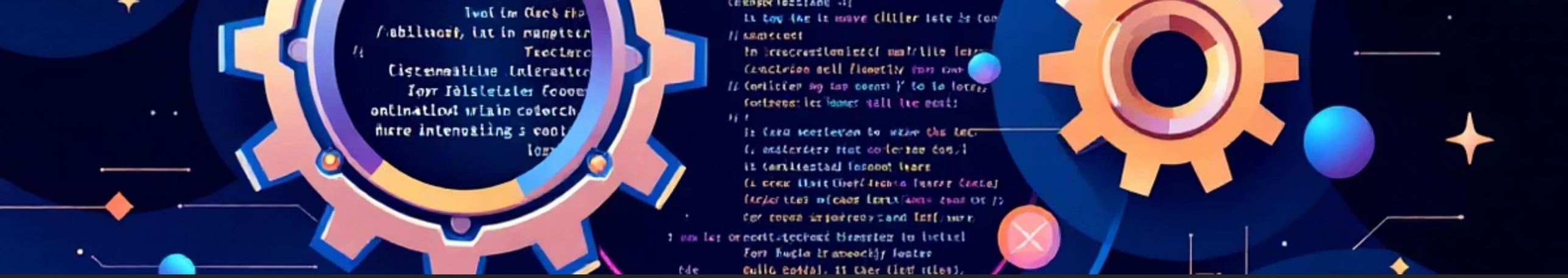
File Watcher & Executor

Go ve Python Tabanlı CI/CD Otomasyon Aracı

Hazırlayan: Eray Turan
2420191037
Bilişim Güvenliği Teknolojisi



PROBLEM VE ÇÖZÜM

Geliştirme Sürecindeki Darboğaz

Karşılaşılan Problem

Yazılım geliştirme sürecinde her kod değişikliğinde (CTRL+S) terminale geçiş yapmak, komutları manuel olarak çalıştırmak hem zaman kaybına neden oluyor hem de odak dağınıklığı yaratıyordu.

Bu süreç özellikle sık test gerektiren projelerde verimliliği düşürüyor ve geliştirici deneyimini olumsuz etkiliyor.

Geliştirilen Çözüm

Dosya sistemini **Kernel seviyesinde** izleyen akıllı bir gözcü (Watcher) mekanizması tasarladık.

Sistem, kod değişikliklerini milisaniyeler içinde algılayarak testleri otomatik olarak başlatıyor ve geliştirme akışını kesintisiz hale getiriyor.

TEKNOLOJI YIĞINI

Kullanılan Teknolojiler

Go (Golang)

Yüksek performans ve düşük kaynak tüketimi sayesinde Watcher modülünün temelini oluşturuyor.

- Hızlı derleme süresi
- Eşzamanlı (concurrent) işlem desteği
- Sistem kaynaklarını verimli kullanım

Python

Otomatik çalıştırılan hedef proje dili olarak seçildi.

- Geniş test framework desteği
- Kolay entegrasyon
- Hızlı prototipleme

fsnotify Kütüphanesi

İşletim sistemi seviyesinde dosya olaylarını (Create, Write, Modify) yakalayan güçlü bir Go kütüphanesi. Cross-platform uyumluluğu ile Windows, Linux ve macOS'ta sorunsuz çalışır.

JSON Yapılandırma

Proje ayarlarını standartlaştırmak ve kullanıcı konfigürasyonlarını yönetmek için JSON formatı kullanıldı. Bu sayede farklı projeler için esnek ayarlar tanımlanabiliyor.

☆ ÖZELLIKLER

Öne Çıkan Özellikler

Projeyi güvenilir, verimli ve kullanıcı dostu kılan kritik özelliklerin detaylı incelemesi:

01

Self-Check (Oto-Kontrol)

Sistem başlangıcında kendi dosya bütünlüğünü doğrular, gerekli dizinlerin varlığını kontrol eder ve Windows/Linux ortam uyumluluğunu test eder. Hata durumunda kullanıcıya anlaşılır mesajlar sunar.

02

Akıllı Filtreleme

Sadece Python (.py) dosyalarını izleyerek gereksiz sistem yükünü önler. Log, cache ve geçici dosyaları (.log, .txt, .tmp) otomatik olarak görmezden gelir ve performansı optimize eder.

03

Debounce Mekanizması

Hızlı ardışık kaydetme işlemlerinde sistemin çökmesini engellemek için 500ms akıllı bekleme süresi uygulanır. Bu sayede IDE'nin otomatik kaydetme özelliği ile çakışma yaşanmaz.

04

İzole Mimari

Kaynak kod ve çalışma alanı klasörleri birbirinden tamamen ayrıştırılmıştır. Bu yapı sayesinde sonsuz döngü (infinite loop) senaryoları önlenir ve sistem kararlı çalışır.

Sonuç ve Kazanımlar

3x

Hız Artışı

Geliştirme sürecinde manuel adımların otomasyonu ile 3 kat hızlanma sağlandı

100%

Hata Azaltma

Manuel test çalıştırma hatalarının tamamen ortadan kaldırılması

500ms

Tepki Süresi

Kod değişikliklerinin algılanması ve testlerin başlatılması arasındaki ortalama süre

Modern CI/CD Deneyimi

Sürekli Entegrasyon (Continuous Integration) mantığı simüle edilerek profesyonel yazılım geliştirme pratiklerine uygun bir araç geliştirildi.

Geliştirici Deneyimi

Manuel işlem yükü azaltılarak geliştiricilerin yaratıcı işlere odaklanması sağlandı. Kesintisiz akış ile verimlilik maksimize edildi.

Teşekkürler!