Mustafa DİNÇ

Yazılım Mühendisi

Baykar Teknoloji

Sayın Yetkili,

Milli Teknoloji Hamlesi'nin öncüsü olan Baykar'da, Merkezi Kontrol Yazılımları biriminizde stajyer olarak yer alma hedefim doğrultusunda başvurumu sunuyorum. İlanınızda belirttiğiniz Android Uygulama Geliştirme (Kotlin/Java), Veritabanı Yönetimi (SQL) ve Analitik Düşünme yetkinlikleri, Github profilimde sunduğum projelerle ve teknik geçmişimle birebir örtüşmektedir.

Yazılım geliştirme sürecine sadece kod yazmak olarak değil, güvenli ve sürdürülebilir sistemler kurmak olarak bakıyorum. Geliştirdiğim "SMS Firewall" projesinde, Android ekosisteminin en kritik bileşenlerini (BroadcastReceiver, Services) kullanarak, binlerce spam mesajı engelleyebilen yüksek performanslı bir güvenlik mimarisi kurdum. Bu projede Kotlin Coroutines ile asenkron veri işleme ve Room Database ile yerel veritabanı yönetimi konularında derinlemesine tecrübe kazandım. İlanınızda özellikle vurguladığınız mobil yazılım ve veritabanı entegrasyonu konusundaki yetkinliğimi bu proje ile somutlaştırdım.

Sadece mobil tarafta değil, kurumsal çözüm süreçlerinde de aktif rol alabilecek yetkinliğe sahibim. C# ve MySQL kullanarak geliştirdiğim "Ticket Repo System", karmaşık SQL sorguları ve veri raporlama (Excel Export) yetenekleriyle, bir IT departmanının operasyonel yükünü hafifletmek üzere tasarlandı. Bu projem, mevcut yazılım çözümlerini analiz etme ve geliştirme konusundaki analitik yaklaşımımın bir göstergesidir.

Ayrıca, modern Web teknolojilerine (React.js) olan hakimiyetim ve görüntü işleme (OpenCV) üzerine yaptığım çalışmalar, ilanınızda "tercih sebebi" olarak belirtilen yeni teknolojilere (Flutter, React Native vb.) adaptasyonumun ne kadar hızlı olacağının kanıtıdır.

Baykar'ın geliştirdiği teknolojilerin arkasındaki o disiplinli ve tutkulu ekibin bir parçası olmayı, teknik birikimimi projelerinize değer katmak için kullanmayı çok istiyorum.

Değerlendirmeniz için teşekkür eder, tanışmak dileğiyle saygılarımı sunarım.

Ankara, Türkiye · +90 (530)-123-6701 · mustafa.din066@gmail.com

Projeler

SMS Firewall & Spam Blocker | *Kotlin, Android SDK, Room DB*
Bu proje, standart bir mesajlaşma uygulamasının ötesine geçerek, Android işletim sisteminin telekomünikasyon katmanına (Telephony Layer) entegre olan ve gelen veri trafiğini gerçek zamanlı analiz eden kapsamlı bir mobil güvenlik modülüdür. Amaç; dolandırıcılık ve spam mesajlarını, bildirim ekranına düşmeden "havada" yakalayıp izole etmektir.

1. Sinyal Yakalama ve Asenkron İşleme (Interception & Concurrency):
Sistemin en kritik bileşeni, işletim seviyesindeki `SMS_DELIVER_ACTION` yayınını dinleyen özelleştirilmiş `BroadcastReceiver` yapısıdır. Receiver'ların kısıtlı yaşam döngüsünü aşmak için `goAsync()` mekanizmasını devreye soktum. Milisaniyeler içinde tamamlanması gereken sinyal işleme sürecini, `Kotlin Coroutines` ve `Dispatchers.IO` thread havuzunu kullanarak asenkron hale getirdim. Bu sayede, yoğun mesaj trafiğinde bile uygulama "ANR" (Yanıt Vermiyor) hatasına düşmeden çalışır.

2. Veritabanı Mimarisi ve Filtreleme:
Veri tutarlılığını sağlamak için `Room Database` kullandım. SQLite üzerine inşa edilen bu yapıda; yasaklı kelimeler ve karantinaya alınan mesajlar için ilişkisel tablolar tasarladım. Gelen her mesajın PDU verisi, bellekteki yasaklı listeyle karşılaştırılır. Tehdit içeren mesajlar, Android'in varsayılan `Inbox`'ına yazılmadan doğrudan şifreli spam veritabanına hapsedilir. Temiz mesajlar ise `ContentResolver API` ile sisteme kaydedilir.

3. Güvenlik İzinleri ve Sistem Entegrasyonu:
Android 10+ güvenlik kısıtlamalarını aşmak için `RoleManager API` kullanarak uygulamanın "Varsayılan SMS Yöneticisi" yetkilerini dinamik olarak devralmasını sağladım. Bu yetki ile `Telephony.Sms.Inbox` üzerinde tam hakimiyet kuruldu. Ayrıca `HeadlessSmsSendService` gibi servisler entegre edilerek tam fonksiyonel bir SMS istemcisi yapısı oluşturuldu.

Kaynak Kod: github.com/conny0506/SMSFirewall

Ticket & Support Repo System | *C#, .NET, MySQL, ADO.NET, Layered Arch.*
Bu proje, IT departmanlarının operasyonel yükünü hafifletmek ve destek süreçlerini ölçülebilir hale getirmek için geliştirilmiş, **C#** ve **.NET** tabanlı bir masaüstü otomasyon yazılımıdır.

1. Mimari Yaklaşım ve Katmanlı Yapı:
"Separation of Concerns" prensibine sadık kalarak; veritabanı bağlantılarını (`MySQLConnectionHelper`), veri erişim katmanını (`TicketRepository`) ve raporlama servisini (`ExcelExporter`) birbirinden izole ettim. Bu modüler yapı, bakım maliyetini düşürmekte ve kodun test edilebilirliğini artırmaktadır.

2. SQL Mühendisliği ve ADO.NET:
ORM araçları yerine, performans optimizasyonu ve sorgu hakimiyeti için **ADO.NET** ve Raw SQL tercih ettim. `TicketRepository` sınıfında, verileri birleştirirken `LEFT JOIN` yapısını kullandım. Bu sayede, henüz çözümü girilmemiş (null) kayıtların raporda kaybolmasını engelledim. Veri bütünlüğü için DBNull kontrollerini `SafeGetDateTime` gibi yardımcı metotlarla kod tarafında yönettim.

3. Veri İşleme ve Raporlama:
Ham veriyi anlamlı bilgiye dönüştürmek için `Tickets` sınıfında sanal property'ler tanımladım. Örneğin, `SolveDate` ve `OpenDate` farkını hesaplayarak `ResolutionTime` (Çözüm Süresi) verisini bellekte dinamik olarak ürettim. İşlenen bu veriler, **ClosedXML** kütüphanesi ile yöneticiler için Excel formatında dışa aktarılmaktadır.

Kaynak Kod: github.com/conny0506/Ticket_Repo_System

CloudGame Frontend Platform | *React.js, Context API, Bootstrap, SPA Arch.*
Bu proje, bulut tabanlı oyun servisleri için modern bir arayüz simülasyonudur. Kullanıcıya kesintisiz bir deneyim sunmak amacıyla **SPA** (Single Page Application) mimarisinde geliştirilmiştir.

1. Component Bazlı Mimari ve React Hooks:
Kodun yönetilebilirliğini artırmak için `GameCard`, `SideMenu` gibi izole bileşenler tasarlandı. Yaşam döngüsü yönetimi için `useEffect` ve `useState` hook'ları kullanılarak, API çağrıları ve state güncellemeleri optimize edildi.

2. Global State Yönetimi (Context API):
"Prop Drilling" sorununu çözmek için harici kütüphaneler yerine React'in yerleşik **Context API** çözümü kullanıldı. Oluşturulan `AppContext` yapısı sayesinde; oturum bilgisi ve alışveriş sepeti gibi global verilere uygulamanın her noktasından erişim sağlandı.

3. Dinamik Yönlendirme ve Asenkron Veri:
React Router ile istemci taraflı (client-side) yönlendirme kurgulandı; sayfalar yeniden yüklenmeden sadece DOM elemanları güncellenerek "Native App" hissi oluşturuldu. Veri akışı için **Axios** kullanılarak RESTful servislere asenkron istekler atıldı.

Kaynak Kod: https://github.com/conny0506/CloudFrontend