Yotech Teknik Dokümantasyon

1. Genel Bakış

Yotech, zincir mağazalar ve bölgesel operasyonlar için tasarlanmış, modüler ve çok kiracılı (multi-tenant) bir perakende yönetim platformudur. SKT takibi, personel giriş-çıkış, form doldurma, vardiya yönetimi, izin talepleri, performans analizleri ve şubeler arası işlemleri tek sistemde toplar. Her firma, yalnızca ihtiyaç duyduğu modülleri aktif hale getirip kullanabilir.

Desteklenen Platformlar

- **Mobil (Flutter):** Android, Zebra el terminalleri ve ilerleyen dönemde iOS
- **Web Panelleri (Next.js):**
- **Grand Admin Paneli:** Yotech ekibi için tüm firmaları izleme, teknik destek ve yönetim
- **Firma Admin Paneli:** Şirket bazlı şube, personel ve istatistik yönetimi
- **Şube Paneli:** Şube yöneticilerinin kendi personel ve operasyonlarını yönetimi

Temel Özellikler

- SKT Takibi (barkod, son kullanma tarihi, alarm)
- Giriş–Çıkış ve Mola Takip
- Vardiya Planlama
- İzin Talepleri ve Onay Süreci
- Sağlık Raporu Yükleme
- Depolar Arası Ürün Transferi
- Arıza Kayıt Yönetimi
- Problemli Ürün Bildirimi
- Performans Değerlendirme (Çalışan + Şube)
- Mağaza İçi Eksik Ürün Takibi
- Görev Yönetimi
- Maaş Bordro Paylaşımı ve Duyurular
- Gerçek Zamanlı Bildirim Sistemi

---

2. Sistem Mimarisi

Backend Teknolojileri

```
Node.js + NestJS (Modüler, kurumsal mimari)
PostgreSQL (Çok kiracılı veritabanı)
Redis (Cache, Queue, Real-time işlemler)
MinIO/S3 (Dosya depolama)
Bull (Job Queue - alarm, bildirim)
Socket.io (Gerçek zamanlı iletişim)
```

Neden NestJS? TypeScript tabanlıdır, modüler mimariyi destekler, mikroservislere kolayca genişletilebilir, bağımlılık enjeksiyonu sağlar ve Swagger ile otomatik API dokümantasyonu üretir.

Mobil Teknolojiler

```
Flutter + Dart
- Tek kod tabanı (Android + iOS + Windows)
- Zebra DataWedge entegrasyonu
- Offline-first mimari (Drift - SQLite ORM)
- State yönetimi: Riverpod / Bloc
- Ağ katmanı: Dio + Retrofit yapısı
```

Web Panel Teknolojileri

```
Next.js 15 + TypeScript
Tailwind + Shadcn/UI
TanStack Query (server state)
Zustand (client state)
React Hook Form + Zod (form validasyonu)
Recharts (grafik ve istatistik)
```

---

3. Veritabanı Şeması (Supabase / PostgreSQL)

Çok Kiracılı (Multi-Tenant) Yapı

Her firma (tenant) benzersiz bir `tenant_id` ile izole edilir. Tüm tablolar `tenant_id` üzerinden ilişkilendirilir. RLS (Row Level Security) sayesinde kullanıcılar yalnızca kendi firmasına ait verileri görebilir.

Ana Tablolar

Tablo	Amaç
tenants	Firma kayıtları, modül erişimleri, SAP entegrasyonu
regions	Firma içi bölge tanımları
branches	Şube bilgileri, konum, geofence ayarları
users	Supabase auth ile entegre kullanıcılar (rol, pozisyon, şube)
products	Ürün temel verisi (barkod, kategori, marka, tedarikçi)
skt_records	SKT kayıtları, alarm ve bildirim sistemi
attendance	Giriş-çıkış kayıtları, GPS bilgileri
break_logs	Mola kayıtları
form_templates / submissions	Dinamik formlar ve doldurulmuş sonuçlar
shifts	Haftalık vardiya çizelgeleri (JSONB)
leave_requests	İzin talepleri, türleri ve durumları

Tablo	Amaç
health_reports	Yüklenen sağlık raporları
inventory_transfers	Şubeler arası ürün transferleri
malfunction_reports	Arıza kayıtları (kategori bazlı)
product_issues	Problemli ürün bildirimleri (SKT + parti numarası)
employee_scores / branch_scores	Performans verileri
stockout_lists / items	Mağaza içi eksik ürün listeleri
tasks / task_assignees / task_items	Görev yönetimi yapısı
payrolls	Maaş bordrolarının paylaşımı
announcements / announcement_reads	Duyurular ve okuma kayıtları
notifications	Kullanıcı bazlı bildirimler

Enum Tipleri

Roller ve sistem durumları PostgreSQL ENUM olarak tanımlanmıştır: - `user_role`: grand_admin, firma_admin, bolge_muduru, sube_muduru, personel - `leave_status`, `leave_type` - `malfunction_priority`, `malfunction_status`, `malfunction_category` - `product_issue_status`, `transfer_status`, `task_status`

---

4. Row Level Security (RLS)

İzolasyon Prensibi

Tüm ana tablolarda RLS **aktiftir**. Her satır `tenant_id = current_tenant_id()` koşuluyla filtrelenir, böylece firmalar arası veri karışımı engellenir.

JWT Entegrasyonu

JWT token içinde kimlik bilgileri şu şekilde taşınır:

```
{
  "sub": "user_uuid",
  "tenant_id": "company_uuid",
  "role": "firma_admin"
}
```

`current_tenant_id()` ve `current_user_role()` fonksiyonları bu bilgileri okur.

Erişim Politikaları

- **Grand Admin:** Tüm veriler üzerinde tam erişim

- **Firma Kullanıcıları:** Kendi tenant verilerine erişim
- **Kullanıcı Kendisi:** Kendi kayıtlarını görebilir (örnek: maaş bordrosu)
- **Şube/Bölge Filtreleme:** Kullanıcının bağlı olduğu alan üzerinden filtreleme

---

5. Entegrasyonlar ve Dış Sistemler

SAP / Mikro Entegrasyonu

- `sap_entegrasyon_aktif` ile aktif hale getirilebilir
- `sap_api_url` ve `sap_api_key` aracılığıyla REST tabanlı veri senkronizasyonu
- Personel ID eşlemesi SAP çalışan koduyla yapılabilir

Dosya Depolama

- MinIO veya Supabase Storage kullanılır:
- SKT ürün fotoğrafları
- Sağlık raporları
- Bordro PDF dosyaları

---

6. Performans ve İndeksleme

İndeksleme Stratejisi

- Tüm kritik tablolarda `(tenant_id, branch_id)` indeksleri bulunur
- SKT ve giriş-çıkış için tarih bazlı indeksler
- Aktif kayıtlar için kısmi indeksler (`aktif = true`)

Önbellekleme ve Arka Plan İşlemleri

- Redis + Bull: Bildirim ve alarm işlemleri kuyruk yapısında
- `check_skt_alarms()` fonksiyonu: SKT yaklaşan ürünler için otomatik bildirim üretir

---

7. Dağıtım (Deployment) ve CI/CD

Supabase + Vercel + AppCenter

- **Supabase:** Veritabanı + kimlik doğrulama + dosya depolama
- **Vercel:** Web panelleri barındırma (Next.js)
- **AppCenter / Play Console:** Flutter mobil dağıtımı

Pipeline Akışı

1. Kod gönderimi → GitHub Actions → Supabase migration (SQL)
2. Web otomatik dağıtımı (Vercel)
3. Mobil uygulama derleme ve yayınlama (CI/CD)

---

8. Gelecek Geliştirmeler

- Yapay zekâ destekli stok tahmini ve satış analizleri
- Offline veri senkronizasyon sistemi
- Otomatik vardiya optimizasyonu
- Modül bazlı dinamik yetkilendirme
- Tüm sistem için GraphQL API geçidi

Doküman Versiyonu: 1.0