TV SERVİS YÖNETİM SİSTEMİ

TEKNİK KULLANIM KILAVUZU

Versiyon: 1.0
Tarih: Aralık 2024
Framework: Hono + TypeScript
Platform: Cloudflare Workers/Pages
Veritabanı: Cloudflare D1 SQLite
Repository: https://github.com/username/webapp

1. SİSTEM MİMARİSİ

1.1 Teknoloji Yığını

- ⚙ **Backend Framework:** Hono v4.x
- ⚙ **Runtime:** Cloudflare Workers
- ⚙ **Veritabanı:** Cloudflare D1 SQLite
- ⚙ **Authentication:** JWT Tokens
- ⚙ **Payment Gateway:** PayTR
- ⚙ **Frontend:** Vanilla JavaScript + TailwindCSS
- ⚙ **Deployment:** Cloudflare Pages
- ⚙ **Monitoring:** Custom Health Dashboard

1.2 Proje Yapısı

```
webapp/ ├── src/ | ├── index.tsx # Ana uygulama dosyası | ├── utils/ | | ├── auth.ts # JWT ve şifre işlemleri | | ├── adminAuth.ts # Admin kimlik doğrulama | | ├── paytr.ts # PayTR entegrasyonu | | ├── logger.ts # Gelişmiş log sistemi | | ├──
```

```
database.ts # Veritabanı optimizasyonu │ │ ├── monitoring.ts # Performans izleme │ │ └── notifications.ts # Email bildirim sistemi │ └── middleware/ │ ├── validation.ts # Giriş doğrulama │ └── errorHandler.ts # Hata yönetimi ├── migrations/ # Veritabanı migration'ları ├── public/static/ # Frontend dosyaları ├── docs/ # Dokümantasyon └── wrangler.jsonc # Cloudflare konfigürasyonu
```

1.3 Veritabanı Şeması

```
-- İşler tablosu CREATE TABLE isler ( id INTEGER PRIMARY KEY AUTOINCREMENT, baslik TEXT NOT NULL, aciklama TEXT, il TEXT NOT NULL, fiyat INTEGER NOT NULL, musteri_adi TEXT, musteri_telefon TEXT, musteri_adres TEXT, durum TEXT DEFAULT 'bekliyor', satin_alan_id INTEGER, olusturulma_tarihi DATETIME DEFAULT CURRENT_TIMESTAMP ); -- Bayiler tablosu CREATE TABLE bayiler ( id INTEGER PRIMARY KEY AUTOINCREMENT, email TEXT UNIQUE NOT NULL, sifre TEXT NOT NULL, ad TEXT NOT NULL, il TEXT NOT NULL, kredi_bakiyesi INTEGER DEFAULT 0, olusturulma_tarihi DATETIME DEFAULT CURRENT_TIMESTAMP ); -- Ödemeler tablosu CREATE TABLE odemeler ( id INTEGER PRIMARY KEY AUTOINCREMENT, bayi_id INTEGER NOT NULL, tutar INTEGER NOT NULL, tip TEXT NOT NULL, durum TEXT DEFAULT 'bekliyor', havale_makbuzu TEXT, admin_notu TEXT, olusturulma_tarihi DATETIME DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (bayi_id) REFERENCES bayiler(id) ); -- Admin kullanıcıları tablosu CREATE TABLE admin_users ( id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE NOT NULL, password TEXT NOT NULL, created_at DATETIME DEFAULT CURRENT_TIMESTAMP );
```

2. API ENDPOINT'LERİ

2.1 Bayi Authentication

POST /api/bayi/register

Açıklama: Yeni bayi kaydı

```
{ "email": "bayi@example.com", "sifre": "guclu_sifre", "ad": "Bayi Adı", "il": "Istanbul" }
```

POST /api/bayi/login

Açıklama: Bayi girişi

```
{ "email": "bayi@example.com", "sifre": "sifre" }
```

Dönüş: JWT token

2.2 İş Yönetimi

GET /api/isler/mevcut

Açıklama: Bayinin ilindeki mevcut işleri listeler

Headers: Authorization: Bearer {token}

GET /api/isler/satin-alinan

Açıklama: Bayinin satın aldığı işleri listeler

Headers: Authorization: Bearer {token}

POST /api/isler/satin-al/:id

Açıklama: İş satın alma (race condition korumalı)

Headers: Authorization: Bearer {token}

2.3 Ödeme İşlemleri

POST /api/odeme/kredi-karti

Açıklama: PayTR ile kredi kartı ödemesi

```
{ "tutar": 100 }
```

POST /api/odeme/banka-havalesi

Açıklama: Banka havalesi bildirimi

```
{ "tutar": 100, "havale_makbuzu": "base64_makbuz_data" }
```

POST /api/paytr/callback

Açıklama: PayTR ödeme callback'i

Not: PayTR tarafından otomatik çağrılır

2.4 Admin Endpoints

GET /api/admin/bekleyen-odemeler

Açıklama: Onay bekleyen ödemeleri listeler

Headers: Authorization: Bearer {admin_token}

POST /api/admin/odeme-onayla/:id

Açıklama: Ödemeyi onaylar

```
{ "admin_notu": "Ödeme onaylandı" }
```

2.5 N8N Webhook

POST /api/n8n/webhook

Açıklama: N8N'den gelen iş verilerini işler

```
{ "baslik": "TV Ekran Değişimi", "aciklama": "55 inç TV
ekranı", "il": "Istanbul", "fiyat": 50, "musteri_adi":
"Ahmet Yılmaz", "musteri_telefon": "05001234567",
"musteri_adres": "Kadıköy/İstanbul" }
```

2.6 Monitoring & Health Check

GET /api/health

Açıklama: Sistem sağlık durumu (JSON)

GET /dashboard

Açıklama: Görsel sistem izleme dashboard'u

3. GELİŞTİRME ORTAMI KURULUMU

3.1 Gereksinimler

- ⚙ Node.js v18+
- ⚙ npm veya yarn
- ⚙ Cloudflare hesabı
- ⚙ Wrangler CLI

3.2 Kurulum Adımları

```
# Repository'yi klonla git clone
https://github.com/username/webapp.git cd webapp #
Bağımlılıkları yükle npm install # Wrangler'ı global olarak
yükle npm install -g wrangler # Cloudflare'e giriş yap wrangler
login # D1 veritabanı oluştur wrangler d1 create webapp-
production # Migration'ları çalıştır wrangler d1 migrations
apply webapp-production --local # Local development başlat npm
run build npm run dev
```

3.3 Ortam Değişkenleri

```
# .dev.vars dosyası JWT_SECRET=your-jwt-secret-key
ADMIN_JWT_SECRET=your-admin-jwt-secret PAYTR_MERCHANT_ID=your-
paytr-merchant-id PAYTR_MERCHANT_KEY=your-paytr-merchant-key
PAYTR_MERCHANT_SALT=your-paytr-merchant-salt
EMAIL_API_KEY=your-email-api-key
EMAIL_FROM=noreply@yourdomain.com
```

4. DEPLOYMENT

4.1 Cloudflare Pages Deployment

```
# Projeyi build et npm run build # Cloudflare Pages'e deploy et wrangler pages deploy dist --project-name webapp # Production migration'ları çalıştır wrangler d1 migrations apply webapp-production
```

4.2 Secrets Yönetimi

```
# Production secrets'ları ayarla wrangler pages secret put JWT_SECRET --project-name webapp wrangler pages secret put ADMIN_JWT_SECRET --project-name webapp wrangler pages secret put PAYTR_MERCHANT_ID --project-name webapp wrangler pages secret put PAYTR_MERCHANT_KEY --project-name webapp wrangler pages secret put PAYTR_MERCHANT_SALT --project-name webapp
```

5. VERİTABANI YÖNETİMİ

5.1 Migration Sistemi

```
# Yeni migration oluştur echo "ALTER TABLE bayiler ADD COLUMN telefon TEXT;" > migrations/0004_add_phone.sql # Local'da uygula wrangler d1 migrations apply webapp-production --local # Production'da uygula wrangler d1 migrations apply webapp-production
```

5.2 Veritabanı Backup

```
# Local veritabanını backup al wrangler d1 execute webapp-production --local --command=".backup backup.db" # Production backup (SQL dump) wrangler d1 execute webapp-production --command=".dump" > backup.sql
```

6. GÜVENLİK

6.1 Authentication & Authorization

Kritik Güvenlik Önlemleri:

- JWT secret'ları güçlü ve unique olmalı
- Şifreler bcrypt ile hash'lenmeli (şu anda geçici bypass var)
- Rate limiting aktif
- Input validation her endpoint'te uygulanmalı
- HTTPS zorunlu (Cloudflare otomatik sağlar)

6.2 Race Condition Koruması

```
// İş satın alma race condition koruması const checkJob = await db.prepare( "SELECT * FROM isler WHERE id = ? AND durum = 'bekliyor'" ).bind(jobId).first(); if (!checkJob) { throw new Error('İş bulunamadı veya zaten satıldı'); } // Double-check locking pattern kullanılıyor
```

7. MONITORING VE LOGGING

7.1 Log Seviyeleri

Level	Açıklama	Örnek
ERROR	Sistem hataları	Database bağlantı hatası
WARN	Potansiyel sorunlar	Yavaş response time
INFO	Genel bilgi	Başarılı ödeme
DEBUG	Detaylı debug bilgisi	API çağrı detayları

7.2 Performance Metrikleri

```
// Monitoring middleware örneği app.use('/api/*', async (c, next) => { const start = Date.now(); await next(); const duration = Date.now() - start; logger.info('API_CALL', { method: c.req.method, url: c.req.url, status: c.res.status, duration: `${duration}ms` }); });
```

8. TESTİNG

8.1 API Testing

```
# Bayi login test curl -X POST
http://localhost:3000/api/bayi/login \ -H "Content-Type:
application/json" \ -d
'{"email":"test@test.com","sifre":"test123"}' # İş listesi test
(token gerekli) curl -X GET
http://localhost:3000/api/isler/mevcut \ -H "Authorization:
Bearer YOUR_JWT_TOKEN" # Health check test curl
http://localhost:3000/api/health
```

8.2 Load Testing

```
# Artillery ile load test npm install -g artillery artillery
quick --count 100 --num 10 http://localhost:3000/api/health
```

9. SORUN GİDERME

9.1 Yaygın Sorunlar

Sorun	Sebep	Çözüm
D1 bağlantı hatası	Yanlış binding	wrangler.jsonc'yi kontrol et
JWT verification failed	Yanlış secret	Environment variable'ları kontrol et
PayTR callback çalışmıyor	Hash mismatch	Merchant key/salt kontrolü
CORS hatası	Yanlış origin	CORS middleware ayarları

9.2 Debug Komutları

```
# Wrangler logs wrangler pages deployment tail --project-name
webapp # Local D1 console wrangler d1 execute webapp-production
--local --command="SELECT * FROM bayiler;" # Health check
detaylı curl -v http://localhost:3000/api/health | jq
```

10. GELİŞTİRME ROADMAP

10.1 Öncelikli İyileştirmeler

- ⚙ **Bcrypt Entegrasyonu:** Gerçek şifre hashleme
- ⚙ **Email Service:** Sendgrid/Mailgun entegrasyonu
- ⚙ **Push Notifications:** Real-time bildirimler
- ⚙ **File Upload:** R2 ile dosya yükleme
- ⚙ **Advanced Analytics:** Detaylı raporlama

10.2 Potansiyel Yeni Özellikler

- ⚙ **Mobile App:** React Native uygulama
- ⚙ **Real-time Chat:** Bayi-müşteri iletişim
- ⚙ **GPS Tracking:** Servis takibi
- ⚙ **Invoice System:** Otomatik fatura oluşturma
- ⚙ **Multi-tenancy:** Şirket bazlı ayrım

11. PERFORMANS OPTİMİZASYONU

11.1 Database Optimizasyonu

```
# Index'ler CREATE INDEX idx_isler_il ON isler(il); CREATE
INDEX idx_isler_durum ON isler(durum); CREATE INDEX
idx_bayiler_email ON bayiler(email); CREATE INDEX
idx_odemeler_bayi_id ON odemeler(bayi_id);
```

11.2 Caching Stratejisi

```
// KV Cache örneği (gelecek implementasyon) const cacheKey =
`jobs:${il}:${Date.now().toString().slice(0, -4)}`; const
cached = await env.KV.get(cacheKey); if (cached) { return
JSON.parse(cached); } // Cache miss, fetch from DB and cache
const jobs = await fetchJobsFromDB(il); await
env.KV.put(cacheKey, JSON.stringify(jobs), { expirationTtl: 60
}); return jobs;
```

Önemli Not: Bu dokümantasyon sistemin mevcut durumunu yansıtır. Geliştirme sürecinde değişiklikler olabilir. Her deployment sonrası güncellemeyi unutmayın.
