

Mini CRM Projesi - Geliştirme Sürecim ve Teknoloji Seçimlerim

Geliştirici: Elif Çetin

Proje: Mini CRM - Müşteri İlişkileri Yönetim Sistemi

Tarih: 2025

Proje Özeti

Bu proje, modern web teknolojileri kullanarak geliştirilmiş, küçük işletmeler için tasarlanmış kapsamlı bir CRM (Müşteri İlişkileri Yönetimi) sistemidir. Proje, tam fonksiyonel bir web uygulaması olarak tasarlanmış ve gerçek dünya ihtiyaçlarını karşılayacak şekilde geliştirilmiştir.

Temel Özellikler

- ✓ Kullanıcı kayıt ve giriş sistemi
- ✓ Müşteri yönetimi (ekleme, düzenleme, silme, listeleme)
- ✓ Not sistemi (müşteri bazlı notlar)
- ✓ Güvenli kimlik doğrulama
- ✓ Responsive tasarım
- ✓ Kapsamlı test coverage

Neden Bu Teknolojileri Seçtim?

1. Next.js 14 - Ana Framework

Neden Next.js?

- **Tek proje, iki işlev:** Hem web sitesi (frontend) hem de sunucu (backend) aynı projede
- **Hızlı geliştirme:** Hazır sayfa yapısı ve API sistemi
- **Kolay deploy:** Vercel gibi platformlarda tek tıkla yayınlama
- **SEO dostu:** Arama motorları için optimize edilmiş
- **App Router:** Modern ve performanslı routing sistemi

Alternatifler nelerdi?

- React + Express: İki ayrı proje yönetmek gerekiyordu
- Vue.js: Daha az popüler, iş bulma açısından dezavantaj
- Vanilla JavaScript: Çok fazla kod yazmak gerekiyordu

2. TypeScript - Programlama Dili

Neden TypeScript?

- **Hata önleme:** Kod yazarken hataları yakalama
- **Daha iyi kod yazma:** Hangi veri tipini kullandığını bilme
- **İş piyasası:** Şirketler TypeScript tercih ediyor
- **Geliştirici deneyimi:** Daha iyi kod tamamlama
- **Tip güvenliği:** Runtime hatalarını compile time'da yakalama

Alternatif: JavaScript

- Daha esnek ama hata yapma riski yüksek
- Büyük projelerde karmaşıklık yaratır

3. MongoDB - Veritabanı

Neden MongoDB?

- **NoSQL yapısı:** JSON benzeri esnek veri yapısı
- **Kolay entegrasyon:** Next.js ile mükemmel uyum
- **Ölçeklenebilirlik:** Büyüyen projeler için uygun
- **Cloud desteği:** MongoDB Atlas ile kolay yönetim

- **Hızlı geliştirme:** Şema değişikliklerinde esneklik

Alternatifler:

- PostgreSQL: Güçlü ama kurulum ve yönetim gerektirir
- SQLite: Basit ama ölçeklenebilirlik sınırlı
- MySQL: Kurulum ve sunucu gerektirir

4. JWT - Kimlik Doğrulama

Neden JWT?

- **Güvenli:** Şifreli token sistemi
- **Stateless:** Sunucuda oturum bilgisi saklamaya gerek yok
- **Hızlı:** Her istekte kullanıcıyı tanıma
- **Yaygın:** Çoğu şirket kullanıyor
- **Cross-domain:** Farklı domainler arası kullanım

Alternatif: Session

- Sunucuda oturum bilgisi saklama gerektirir
- Daha karmaşık yönetim

5. TailwindCSS - Tasarım

Neden TailwindCSS?

- **Hızlı tasarım:** Hazır CSS sınıfları
- **Tutarlılık:** Tüm sayfalarda aynı görünüm
- **Responsive:** Mobil ve masaüstü uyumlu
- **Küçük boyut:** Sadece kullandığın CSS'ler dahil edilir
- **Utility-first:** Hızlı prototipleme

Alternatifler:

- Bootstrap: Daha büyük, daha az esnek
- CSS Modules: Daha fazla kod yazma

- Styled Components: React'e bağımlı



Proje Yapısı Neden Bu Şekilde?

Dosya Organizasyonum:

```
src/
├── app/                                # Next.js App Router
│   ├── api/                            # Backend API endpoints
│   │   ├── auth/                       # Kimlik doğrulama API'leri
│   │   ├── customers/                  # Müşteri işlemleri API'leri
│   │   └── notes/                      # Not işlemleri API'leri
│   ├── customers/                      # Müşteri sayfaları
│   ├── login/                          # Giriş sayfası
│   └── register/                       # Kayıt sayfası
├── lib/                                # Yardımcı fonksiyonlar
│   ├── models/                         # Veritabanı modelleri
│   ├── auth.ts                         # Kimlik doğrulama fonksiyonları
│   ├── mongodb.ts                      # Veritabanı bağlantısı
│   └── utils.ts                        # Genel yardımcı fonksiyonlar
├── providers/                          # React context providers
├── types/                              # TypeScript tip tanımları
└── __tests__/                          # Test dosyaları
```

Neden bu yapı?

- **Next.js 14 App Router:** Yeni ve önerilen yapı
- **Kolay bulma:** Her şey kendi yerinde
- **Büyüme:** Yeni özellikler eklemek kolay
- **Separation of Concerns:** Her katmanın kendi sorumluluğu
- **Scalability:** Proje büyüdükçe yapı korunur



Güvenlik Seçimlerim

Şifre Hashleme

- **bcryptjs:** Şifreleri güvenli şekilde saklama
- **Salt:** Her şifreye özel ek güvenlik
- **Cost factor:** Güvenlik ve performans dengesi

JWT Token

- **Expiration:** Token'ların süresi doluyor
- **Secret Key:** Güvenli anahtar kullanımı
- **Refresh Token:** Güvenli token yenileme

Input Validation

- **Sunucu tarafı:** Tüm veriler sunucuda kontrol ediliyor
- **TypeScript:** Veri tiplerini kontrol etme
- **Sanitization:** XSS ve injection saldırılarına karşı koruma

Environment Variables

- **Güvenlik:** Hassas bilgileri kodda saklamama
- **Esneklik:** Farklı ortamlar için farklı ayarlar
- **Best Practices:** .env dosyaları ile yönetim



Kullanıcı Deneyimi Seçimlerim

Responsive Tasarım

- **Mobil öncelikli:** Telefonlarda da çalışır
- **Touch-friendly:** Dokunmatik ekranlar için optimize
- **Breakpoint strategy:** Farklı ekran boyutları için optimize

Loading States

- **Kullanıcı geri bildirimi:** İşlemler sırasında bilgi verme
- **Error handling:** Hataları güzel şekilde gösterme
- **Optimistic updates:** Kullanıcı deneyimini iyileştirme

Navigation

- **Kolay gezinme:** Menüler ve butonlar net
 - **Breadcrumbs:** Nerede olduğunu bilme
 - **Consistent UI:** Tüm sayfalarda tutarlı tasarım
-

Deploy Seçimlerim

Vercel

Neden Vercel?

- **Next.js uyumlu:** En iyi performans
- **Ücretsiz:** Kişisel projeler için yeterli
- **Otomatik deploy:** GitHub'a push ettiğinde otomatik yayınlama
- **Hızlı:** CDN ile dünya çapında hızlı erişim
- **Environment Variables:** Güvenli konfigürasyon yönetimi

Environment Variables

- **Güvenlik:** Hassas bilgileri kodda saklamama
 - **Esneklik:** Farklı ortamlar için farklı ayarlar
 - **Best Practices:** Production ve development ayrımı
-



Test Stratejim

Jest + React Testing Library

Neden bu kombinasyon?

- **Jest:** Güçlü test framework
- **React Testing Library:** Kullanıcı odaklı testler
- **Coverage:** Tüm özellikler test edildi
- **Maintainability:** Testler kodla birlikte güncellenir

Test Kategorileri

1. **Unit Tests:** Bireysel fonksiyon testleri
2. **Integration Tests:** API endpoint testleri
3. **Component Tests:** UI bileşen testleri
4. **E2E Tests:** Kullanıcı senaryoları

Test Best Practices

- **Arrange-Act-Assert:** Test yapısı
- **Mocking:** Dış bağımlılıkları simüle etme
- **Cleanup:** Test sonrası temizlik
- **Descriptive names:** Anlamlı test isimleri



Geliştirme Süreci

1. Planlama ve Analiz

- **İhtiyaç analizi:** Hangi özellikler gerekli?
- **Teknoloji seçimi:** En uygun araçlar neler?
- **Mimari tasarım:** Proje yapısı nasıl olacak?

2. Temel Altyapı

- **Next.js kurulumu:** Proje başlangıcı
- **TypeScript konfigürasyonu:** Tip güvenliği
- **TailwindCSS entegrasyonu:** Styling sistemi
- **MongoDB bağlantısı:** Veritabanı kurulumu

3. Backend Geliştirme

- **API endpoints:** RESTful API tasarımı
- **Veritabanı modelleri:** MongoDB şemaları
- **Kimlik doğrulama:** JWT implementasyonu
- **Validation:** Veri doğrulama sistemi

4. Frontend Geliştirme

- **Sayfa yapısı:** Routing ve layout
- **Bileşenler:** Reusable UI components
- **State management:** Context API kullanımı
- **Responsive design:** Mobil uyumluluk

5. Test ve Kalite

- **Unit testler:** Bireysel fonksiyon testleri
- **Integration testler:** API testleri
- **UI testler:** Bileşen testleri
- **Code review:** Kod kalitesi kontrolü

6. Deploy ve Yayınlama

- **Environment setup:** Production konfigürasyonu
 - **Vercel deploy:** Otomatik yayınlama
 - **Monitoring:** Performans takibi
 - **Documentation:** Kullanım kılavuzu
-

Gelecek Planlarım

Eklenebilecek Özellikler

- **E-posta bildirimleri:** Müşteri takibi
- **Raporlama:** Satış ve müşteri analizi
- **Takvim entegrasyonu:** Toplantı planlama
- **Dosya yükleme:** Müşteri belgeleri
- **Dashboard:** Görsel analitikler
- **Export/Import:** Veri aktarımı

Ölçeklendirme

- **PostgreSQL:** Daha büyük veritabanı
- **Redis:** Hızlı önbellek
- **Microservices:** Modüler yapı
- **Docker:** Containerization
- **CI/CD:** Otomatik deployment pipeline

Performans İyileştirmeleri

- **Image optimization:** Görsel optimizasyonu
- **Code splitting:** Lazy loading
- **Caching:** Önbellek stratejileri
- **CDN:** İçerik dağıtım ağı

Öğrendiğim Dersler

Teknoloji Seçimi

- **İhtiyaça göre seçim:** En popüler değil, en uygun
- **Basitlik:** Karmaşık çözümler yerine basit ve etkili

- **Gelecek odaklı:** Güncel teknolojiler kullanma
- **Ekosistem:** Teknolojilerin birbiriyle uyumu

Proje Yönetimi

- **Adım adım geliştirme:** Her özelliği ayrı ayrı ekleme
- **Test etme:** Her aşamada kontrol etme
- **Dokümantasyon:** Kod açıklamaları ve README
- **Version control:** Git ile düzenli commit'ler

Kullanıcı Odaklılık

- **Basit arayüz:** Karmaşık olmayan tasarım
- **Hızlı yükleme:** Performans önemli
- **Mobil uyumlu:** Her cihazda çalışma
- **Accessibility:** Erişilebilirlik standartları



Kod Kalitesi





- **Clean code:** Okunabilir ve maintainable kod
- **Type safety:** TypeScript ile tip güvenliği
- **Error handling:** Kapsamlı hata yönetimi
- **Code review:** Sürekli iyileştirme

Sonuç

Bu proje, modern web geliştirme teknolojilerini kullanarak, gerçek ihtiyaçları karşılayan bir uygulama oluşturmanın mükemmel bir örneğidir. Her teknoloji seçimi, projenin ihtiyaçlarına ve gelecek planlarına göre yapılmıştır.

Başarılar

-  Tam fonksiyonel CRM sistemi
-  Modern teknoloji stack'i

-  Kapsamlı test coverage
-  Güvenli ve performanslı
-  Responsive ve kullanıcı dostu
-  Production-ready deploy

Öğrenilen Teknolojiler

- Next.js 14 App Router
- TypeScript
- MongoDB
- JWT Authentication
- TailwindCSS
- Jest Testing
- Vercel Deployment

Gelecek Hedefler

- Daha fazla özellik ekleme
- Performans optimizasyonu
- Kullanıcı geri bildirimleri
- Sürekli iyileştirme

Bu proje, modern web geliştirme süreçlerini öğrenmek ve uygulamak için mükemmel bir fırsat oldu. Her adımda yeni şeyler öğrendim ve gerçek dünya problemlerini çözme deneyimi kazandım.

Proje Linki: [GitHub Repository](#)

Canlı Demo: [Vercel Deploy](#)

Dokümantasyon: [README.md](#)

Bu dokümantasyon, projenin geliştirme sürecini, teknoloji seçimlerini ve öğrenilen dersleri kapsamlı bir şekilde açıklamaktadır. Gelecekteki projelerim için referans olarak kullanılabilir.

Bu dokümantasyon Mini CRM projesi için hazırlanmıştır.

Geliştirici: Elif Çetin | Tarih: 10.07.2025