Backend Developer • Case Study

PROJE → **Real-Time Mesajlaşma Sistemi Proje Dokümantasyonu**

Genel Bilgilendirme

Bu proje kapsamında, kullanıcıların birbiriyle gerçek zamanlı mesajlaşabileceği basit bir kullanıcı sistemi geliştirilecektir. Sistem, modern web teknolojileri kullanılarak ölçeklenebilir ve performanslı bir yapıda tasarlanacaktır.

Kullanılacak Teknolojiler

Bu projede aşağıdaki teknolojilerin aktif olarak kullanılması gerekmektedir:

- **Node.js** - Sunucu tarafı JavaScript çalışma ortamı
- **Express.js** - Web uygulama framework'ü
- **MongoDB** - NoSQL veritabanı
- **RabbitMQ** - Mesaj kuyruğu sistemi
- **Redis** - In-memory veri yapısı deposu
- **JWT** - JSON Web Token kimlik doğrulama
- **Socket.IO** - Gerçek zamanlı iletişim kütüphanesi
- **Cron** - Zamanlanmış görev yöneticisi

Proje Gereksinimleri

1. Veri Modelleri

Sistemde aşağıdaki veri modelleri oluşturulacaktır:

- **User** - Kullanıcı bilgileri
- **Conversation** - Konuşma bilgileri
- **Message** - Mesaj bilgileri
- **AutoMessage** - Otomatik mesaj bilgileri

2. Kullanıcı Yönetimi API'leri

I. Kimlik Doğrulama Endpoint'leri

- `POST /api/auth/register` - Yeni kullanıcı kaydı
- `POST /api/auth/login` - Kullanıcı giriş işlemi
- `POST /api/auth/refresh` - Access token yenileme
- `POST /api/auth/logout` - Kullanıcı çıkış işlemi
- `GET /api/user/list` - Sistemdeki kullanıcıları listeleme
- `GET /api/auth/me` - Kullanıcı profil bilgilerini görüntüleme

3. Mesajlaşma API'leri

I. Mesaj Yönetimi Endpoint'leri

- Ürünün tasarım yönlendirmesi
- Ton, renk, UX öncelikleri

4. Socket.IO Event'leri

I. Gerçek Zamanlı İletişim Event'leri

- `connection` - Kullanıcının sisteme bağlanması
- `join_room` - Belirli bir konuşma odasına katılma
- `send_message` - Gerçek zamanlı mesaj gönderme
- `message_received` - Mesaj alındı bildirimi
- `user_online` - Kullanıcının online durumu bildirimi
- `disconnect` - Kullanıcının sistemden ayrılması

Geliştirme Gereksinimleri

1. Authentication Middleware

Güvenli API erişimi için aşağıdaki özellikleri içeren middleware geliştirmeniz gerekmektedir:

- JWT token doğrulama mekanizması
- Authorization header'dan token çıkarma
- Doğrulanmış kullanıcı bilgilerinin request nesnesine ekleme
- Hata durumlarında uygun HTTP status kodları döndürme

2. RabbitMQ Kuyruk Sistemi

Asenkron mesaj işleme için aşağıdaki bileşenleri içeren sistem:

- Otomatik mesajlar için özel kuyruk yapısı
- Message producer (üretici) ve consumer (tüketici) servisleri
- Hata yönetimi ve tekrar deneme (retry) mekanizması

3. Cron Job - Otomatik Mesaj Sistemi

Zamanlanmış görev yönetimi için aşağıdaki özellikleri içeren sistem:

- Her gece saat 02:00'da çalışan otomatik görev
- Rastgele kullanıcı eşleştirme algoritması
- Mesaj içeriği üretimi ve veritabanına kaydetme
- RabbitMQ kuyruğuna mesaj ekleme işlemi

4. Redis Servisleri

Performans optimizasyonu için aşağıdaki cache sistemleri:

- Kullanıcı online/offline durumu takibi
- Konuşma verilerinin cache'lenmesi
- Session yönetimi
- Geçici veri saklama mekanizması

5. Socket.IO Implementasyonu

Gerçek zamanlı iletişim için aşağıdaki özellikleri içeren sistem:

- JWT tabanlı kullanıcı kimlik doğrulaması
- Gerçek zamanlı mesaj gönderim sistemi
- Kullanıcı online durumu takibi
- Oda (room) yönetimi ve kullanıcı gruplama

Sistem Akışı ve Otomasyon

Bu bölümde kullanıcıların gerçekleştirebileceği temel işlemler ve sistemin arka planda otomatik olarak yürüteceği süreçler detaylandırılmıştır.

1. Kullanıcı İşlemleri

I. Kayıt ve Kimlik Doğrulama

Kullanıcılar username, email ve password bilgileri ile sisteme kayıt olabilirler. Başarılı giriş işlemi sonrasında JWT tabanlı Access Token ve Refresh Token alırlar.

II. Profil Yönetimi

Kullanıcılar kimlik doğrulaması gerektiren endpoint'ler üzerinden kendi profil bilgilerini görüntüleyebilir ve güncelleyebilirler.

III. Gerçek Zamanlı Mesajlaşma

- Socket.IO teknolojisi kullanılarak anlık mesaj gönderim ve alma işlemleri
- Karşı tarafın yazma durumunu gösteren typing event'i
- Mesaj okundu bilgisi ve anlık bildirimler

IV. Mesaj Geçmişi Yönetimi

Kullanıcılar önceki konuşmalarını ve bu konuşmalardaki tüm mesajları RESTful API endpoint'leri aracılığıyla görüntüleyebilirler.

V. Güvenli Oturum Sonlandırma

Sistem çıkışı sırasında JWT token'ların geçersiz kılınması ve güvenli oturum sonlandırma işlemleri.

2. Otomatik Sistem Süreçleri

Bu bölüm projenin en kritik kısımlarından biridir ve zamanlanmış görev yönetimi ile asenkron işlem becerilerini test etmeyi amaçlar. Sistem, üç ayrı ve yönetilebilir aşamada çalışır: **Planlama**, **Kuyruğa Alma** ve **İşleme**.

1. Adım: Mesaj Planlama Servisi (Cron Job - Gece 02:00)

Amaç: Aktif kullanıcıları otomatik olarak eşleştirerek gönderilecek mesajları toplu halde hazırlamak.

İşlem Süreci:

- Her gece saat 02:00'da otomatik olarak tetiklenir
- Veritabanındaki tüm aktif kullanıcıları çeker
- Kullanıcı listesini rastgele karıştırır (shuffle algoritması)
- Karıştırılmış listeyi ikişerli gruplara ayırarak (gönderici, alıcı) çiftleri oluşturur
- Her çift için rastgele mesaj içeriği hazırlar
- Gönderim için gelecek tarih (sendDate) belirler
- Tüm bilgileri AutoMessage koleksiyonuna kaydeder

2. Adım: Kuyruk Yönetimi Servisi (Worker Cron Job - Dakikada Bir)

Amaç: Gönderim zamanı gelen mesajları tespit edip RabbitMQ sistemine yönlendirmek.

İşlem Süreci:

- Her dakika otomatik olarak çalışır
- AutoMessage koleksiyonunda sendDate'i geçmiş ve isQueued: false olan mesajları arar
- Tespit edilen mesajları RabbitMQ'daki message_sending_queue kuyruğuna gönderir
- Aynı mesajın tekrar işlenmemesi için AutoMessage kaydını isQueued: true olarak günceller

3. Adım: Mesaj Dağıtım Servisi (RabbitMQ Consumer)

Amaç: Kuyruktaki mesajları işleyerek alıcılara ulaştırmak.

İşlem Süreci:

- message_sending_queue kuyruğunu sürekli dinler
- Kuyruğa gelen görevleri anında alır ve işler
- Görev bilgilerine göre yeni Message dökümanı oluşturur ve veritabanına kaydeder
- Socket.IO üzerinden alıcıya message_received eventi ile anlık bildirim gönderir
- AutoMessage kaydını isSent: true olarak güncelleyerek işlemi tamamlar

Sistem Avantajları:

Bu üç aşamalı yapı sayesinde görevlerin zamanlanması, işleme alınacakların tespiti ve gerçek gönderim işlemlerinin birbirinden ayrıştırılması sağlanır. Bu yaklaşım ölçeklenebilir, hataya dayanıklı ve yönetilebilir bir otomatik mesajlaşma sistemi oluşturur.

3. Online Kullanıcı Takip Sistemi

Bu sistem Socket.IO ve Redis teknolojilerini kullanarak kullanıcıların online durumlarını gerçek zamanlı olarak takip eder.

I. Kullanıcı Bağlantı Yönetimi

Kullanıcılar username, email ve password bilgileri ile sisteme kayıt olabilirler. Başarılı giriş işlemi sonrasında JWT tabanlı Access Token ve Refresh Token alırlar.

Kullanıcı Sisteme Bağlandığında (connection event):

- Socket.IO üzerinden gelen bağlantı JWT token ile doğrulanır
- Kimlik doğrulaması başarılı olan kullanıcı Redis'teki online kullanıcılar listesine (Set veri yapısı) eklenir
- Diğer kullanıcılara kullanıcının online olduğu bilgisi broadcast edilir

Kullanıcı Sistemden Ayrıldığında (disconnect event):

- Bağlantısı kesilen kullanıcının kimliği Redis'teki online listesinden kaldırılır
- Diğer kullanıcılara kullanıcının offline olduğu bilgisi iletilir

II. Online Durum Sorguları

Kullanıcılar username, email ve password bilgileri ile sisteme kayıt olabilirler. Başarılı giriş işlemi sonrasında JWT tabanlı Access Token ve Refresh Token alırlar.

- **Anlık Online Kullanıcı Sayısı:** Redis'teki online kullanıcılar Set'inin eleman sayısı sorgulanarak anlık online kullanıcı sayısı alınır.
- **Belirli Kullanıcının Online Durumu:** Redis Set'inde belirli bir kullanıcı ID'sinin varlığı kontrol edilerek o kullanıcının online/offline durumu tespit edilir.
- **Online Kullanıcı Listesi:** Bir test endpointi ile istatistik amaçlı Redis Set'indeki tüm online kullanıcı ID'leri listelenir.

Değerlendirilecek Noktalar:

1. **Backend Geliştirme:** Node.js ve Express.js ile API geliştirme becerileri
2. **Veritabanı Yönetimi:** MongoDB ile NoSQL veritabanı tasarımı ve işlemleri
3. **Güvenlik:** JWT tabanlı kimlik doğrulama ve yetkilendirme implementasyonu
4. **Gerçek Zamanlı İletişim:** Socket.IO ile real-time mesajlaşma sistemi
5. **Asenkron İşlemler:** RabbitMQ ile message queue sistemi kurulumu
6. **Cache Yönetimi:** Redis ile performans optimizasyonu
7. **Zamanlanmış Görevler:** Cron job'ları ile otomatik sistem süreçleri
8. **Kod Kalitesi:** Temiz kod yazma prensipleri ve best practice'ler

Bonus Kriterler:

Bu özelliklerin kullanılması bonus puan olacaktır

Temel Bonus Özellikler

1. Logger kullanımı (Winston, Pino vs.)
2. Elastic Search kullanımı (Mesaj arama, indeksleme)
3. Swagger/OpenAPI entegrasyonu (API dokümantasyonu)
4. Error Yakalama (Sentry vs.)

Güvenlik ve Performance

1. Rate Limiting middleware
2. Input Validation ve sanitization
3. Database indexing ve optimizasyon
4. Redis caching stratejileri
5. Security headers (Helmet vs.)