🚂 Railway Deployment Rehberi - Workigom Backend

📦 Proje Yapısı

```
workigom/
├── backend/                 # Backend servisi
│   ├── Dockerfile           # Backend için Docker yapılandırması
│   ├── railway.toml         # Backend için Railway yapılandırması
│   ├── src/
│   ├── prisma/
│   └── package.json
├── Dockerfile               # Frontend için Docker yapılandırması
├── railway.json             # Root seviye Railway yapılandırması (backend için)
└── package.json
```

🎯 Backend Deployment (ÖNERİLEN YÖNTEM)

Yöntem 1: Backend Klasöründen Deploy (En Basit)

Bu yöntem **en basit ve önerilen** yöntemdir.

Adım 1: Railway'de Yeni Servis Oluştur

1. Railway Dashboard'a git: https://railway.app
2. Projenize tıklayın
3. **"+ New"** → **"GitHub Repo"** seçin
4. `workigom` repository'sini seçin

Adım 2: Root Directory Ayarla

Railway servisi oluşturulduğunda:

1. **Settings** sekmesine git
2. **Build** bölümünü bul
3. **Root Directory** ayarını şu şekilde değiştir:
   `backend`
4. **Build Command** (otomatik algılanacak):
   `npm ci && npm run prisma:generate && npm run build`
5. **Start Command** (otomatik algılanacak):
   `node dist/server.js`

Adım 3: Environment Variables Ekle

Variables sekmesinde şu değişkenleri ekle:

```
# Database (Railway PostgreSQL eklenmişse otomatik gelir)
DATABASE_URL=${{Postgres.DATABASE_URL}}

# JWT
JWT_SECRET=<güvenli-random-string>
JWT_EXPIRES_IN=7d

# Server
NODE_ENV=production
PORT=3001

# CORS (frontend URL'inizi girin)
FRONTEND_URL=https://your-frontend-url.railway.app

# File Upload
MAX_FILE_SIZE=5242880
UPLOAD_DIR=uploads
```

Adım 4: Deploy

1. **Deploy** butonuna tıkla
2. Build loglarını izle
3. Deploy tamamlandığında URL'yi kopyala

Yöntem 2: Monorepo Yapılandırması (İki Servis)

Hem frontend hem backend'i aynı repo'dan deploy etmek için:

Backend Servisi

1. **Root Directory**: `backend`
2. **Dockerfile Path**: `Dockerfile` (backend klasöründeki)
3. **Start Command**: `node dist/server.js`

Frontend Servisi

1. **Root Directory**: (boş bırak veya `/`)
2. **Dockerfile Path**: `Dockerfile` (root'taki)
3. **Start Command**: (Nginx otomatik başlar)

🔧 Dockerfile Özellikleri

Backend Dockerfile

✅ **Multi-stage build** - Küçük production image
✅ **Prisma generation** - Production'da çalışır
✅ **Health check** - `/api/health` endpoint kontrolü
✅ **Security** - Sadece production dependencies

```
# Build stage
FROM node:20-alpine AS builder
WORKDIR /app
COPY package*.json ./
COPY prisma ./prisma/
RUN npm ci
COPY . .
RUN npm run prisma:generate
RUN npm run build

# Production stage
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY --from=builder /app/prisma ./prisma
RUN npm run prisma:generate
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/uploads ./uploads
RUN mkdir -p uploads/resumes uploads/avatars
EXPOSE 3001
CMD ["node", "dist/server.js"]
```

🗄️ Database Setup

PostgreSQL Ekleme

1. Railway Dashboard'da **"+ New" → "Database" → "PostgreSQL"**
2. PostgreSQL servisi otomatik olarak `DATABASE_URL` environment variable'ı ekler
3. Backend servisinde bu değişken otomatik kullanılacak

Migration Çalıştırma

İlk deploy'dan sonra migration çalıştır:

```
# Railway CLI kullanarak
railway run npm run prisma:migrate:deploy

# Ya da Railway Dashboard'dan One-off Command
npm run prisma:migrate:deploy
```

🚨 Sorun Giderme

Hata: "/app/dist": not found

Sebep: Railway yanlış root directory ile çalışıyor.

Çözüm:
- Settings → Build → Root Directory → `backend` olarak ayarla
- Ya da repository'yi `backend` klasöründen deploy et

Hata: Prisma Client not generated

Sebep: Prisma client production'da generate edilmemiş.

Çözüm: Dockerfile'da zaten `RUN npm run prisma:generate` var. Yeniden deploy dene.

Hata: Port already in use

Sebep: Railway otomatik PORT atıyor.

Çözüm: Backend kodunda `process.env.PORT` kullanıldığından emin ol.

```
const PORT = process.env.PORT || 3001;
```

Hata: CORS hatası

Sebep: Frontend URL'si backend'de whitelist'te yok.

Çözüm: `FRONTEND_URL` environment variable'ını doğru ayarla.

✅ Deployment Checklist

Backend deploy etmeden önce kontrol et:

- [ ] GitHub repository güncel
- [ ] `backend/.env.example` dosyası mevcut
- [ ] Railway'de PostgreSQL servisi eklendi
- [ ] Environment variables ayarlandı
- [ ] Root Directory = `backend` olarak ayarlandı
- [ ] Health check endpoint (`/api/health`) çalışıyor
- [ ] CORS ayarları yapılandırıldı

📊 Monitoring

Railway otomatik olarak sağlar:

- **Logs**: Real-time application logs
- **Metrics**: CPU, Memory, Network kullanımı
- **Deployments**: Deployment geçmişi ve rollback
- **Health Checks**: Otomatik health monitoring

🔄 Sürekli Deployment

Railway otomatik olarak:

- Her `main` branch push'unda yeniden deploy eder
- Pull request'ler için preview environment oluşturur
- Build hatalarında deploy'u durdurur

📱 Railway CLI (Opsiyonel)

Lokal olarak Railway ile çalışmak için:

```
# Railway CLI kur
npm i -g @railway/cli

# Login
railway login

# Projeyi bağla
railway link

# Environment variables'ları çek
railway variables

# Lokal olarak çalıştır Railway environment'ı ile
railway run npm run dev

# Deploy et
railway up
```

🎉 Deploy Sonrası

Backend başarıyla deploy edildikten sonra:

1. **URL'yi kopyala**: `https://workigom-backend-production.up.railway.app`
2. **Health check test et**: `https://your-backend-url/api/health`
3. **Frontend'de backend URL'yi güncelle**
4. **Prisma migration'ları çalıştır**
5. **Seed data ekle** (gerekirse)

💡 İpuçları

1. Hızlı Deploy için Railway CLI

```
cd backend
railway up
```

2. Logs İzleme

```
railway logs
```

3. Database Shell

```
railway connect postgres
```

4. Environment Variables

```
railway variables set KEY=value
```

5. Cost Optimization

- **Sleep Policy**: Kullanılmadığında uyut (Hobby plan)
- **Autoscaling**: Gerektiğinde otomatik scale (Pro plan)

🆘 Destek

Sorun yaşarsan:

1. **Railway Docs**: https://docs.railway.app
2. **Railway Discord**: https://discord.gg/railway
3. **Logs**: Railway Dashboard → Logs sekmesi

Hazırlayan: DeepAgent AI
Tarih: 23 Ekim 2025
Proje: Workigom Backend Deployment