Deployment-Anleitung

Lokale Entwicklung

Start

```
cd /home/ubuntu/nightlife_os

# Alle Apps gleichzeitig starten
pnpm dev

# Oder einzeln
pnpm --filter @nightlife/club-app dev      # http://localhost:3000
pnpm --filter @nightlife/dj-console dev    # http://localhost:3001
```

URLs

- **Club-App:** http://localhost:3000
- **DJ-Console:** http://localhost:3001

Test-Accounts

- **Standard User:** `john@doe.com` / `johndoe123` (Admin)
- **Admin:** `admin@club.internal` / `DFjk1289#3`
- **Admin-Shortcut:** `admin` / `DFjk1289#3`

Production-Deployment

Build

```
pnpm build
```

Dies erstellt optimierte Production-Builds für beide Apps:
- `apps/club-app/.next/`
- `apps/dj-console/.next/`

Start Production Server

```
# Club-App
pnpm --filter @nightlife/club-app start

# DJ-Console
pnpm --filter @nightlife/dj-console start
```

Deployment auf Vercel/Netlify

Vercel (Empfohlen)

1. **Root-Projekt connecten:**
   `bash`
   `vercel`

2. **Projekt-Einstellungen:**
   - Framework: Next.js
   - Build Command: `pnpm build --filter=@nightlife/club-app` (für Club-App)
   - Build Command: `pnpm build --filter=@nightlife/dj-console` (für DJ-Console)
   - Output Directory: `apps/club-app/.next` bzw. `apps/dj-console/.next`

3. **Environment Variables:**
   Keine erforderlich (Firebase-Config wird aus `public/config.js` geladen)

Alternative: Monorepo mit zwei Projekten

Erstelle zwei separate Vercel-Projekte:

Club-App:
- Root Directory: `apps/club-app`
- Build Command: `cd ../.. && pnpm install && pnpm --filter @nightlife/club-app build`
- Output Directory: `.next`

DJ-Console:
- Root Directory: `apps/dj-console`
- Build Command: `cd ../.. && pnpm install && pnpm --filter @nightlife/dj-console build`
- Output Directory: `.next`

Docker-Deployment

Dockerfile (Club-App)

```
FROM node:18-alpine AS base

# Install pnpm
RUN npm install -g pnpm

# Dependencies
FROM base AS deps
WORKDIR /app
COPY package.json pnpm-workspace.yaml pnpm-lock.yaml ./
COPY apps/club-app/package.json ./apps/club-app/
COPY packages/core/package.json ./packages/core/
COPY packages/ui/package.json ./packages/ui/
RUN pnpm install --frozen-lockfile

# Builder
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .
RUN pnpm --filter @nightlife/club-app build

# Runner
FROM base AS runner
WORKDIR /app
ENV NODE_ENV production
COPY --from=builder /app/apps/club-app/.next/standalone ./
COPY --from=builder /app/apps/club-app/.next/static ./apps/club-app/.next/static
COPY --from=builder /app/apps/club-app/public ./apps/club-app/public

EXPOSE 3000
ENV PORT 3000

CMD ["node", "apps/club-app/server.js"]
```

Docker Compose

```yaml
version: '3.8'

services:
  club-app:
    build:
      context: .
      dockerfile: apps/club-app/Dockerfile
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production

  dj-console:
    build:
      context: .
      dockerfile: apps/dj-console/Dockerfile
    ports:
      - "3001:3001"
    environment:
      - NODE_ENV=production
```

Firebase-Setup

Sicherheitsregeln deployen

```
firebase deploy --only firestore:rules
```

Indizes erstellen

```
firebase deploy --only firestore:indexes
```

Storage-Rules (für Bilder)

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read: if request.auth != null;
      allow write: if request.auth != null &&
                    request.resource.size < 5 * 1024 * 1024;
    }
  }
}
```

Performance-Optimierung

1. Next.js-Config

```
// next.config.js
module.exports = {
  compress: true,
  productionBrowserSourceMaps: false,
  optimizeFonts: true,
  images: {
    domains: ['firebasestorage.googleapis.com'],
    formats: ['image/avif', 'image/webp'],
  },
}
```

2. Caching-Strategy

- Static Assets: Infinite Cache
- API Routes: Stale-While-Revalidate
- Firebase Data: Realtime Updates

3. Code-Splitting

- Lazy Loading für Chat-Komponenten
- Dynamic Imports für QR-Scanner
- Route-basiertes Splitting (automatisch)

Monitoring

Empfohlene Tools

- **Vercel Analytics** - Performance Monitoring
- **Firebase Console** - Firestore-Usage
- **Sentry** - Error Tracking

Custom Logging

```
// In Production
console.log = () => {}
console.debug = () => {}
// Nur Errors & Warnings behalten
```

Security-Checklist

- [x] Firebase Security Rules deployed
- [x] HTTPS erzwungen
- [x] DeviceId-Hashing implementiert
- [x] CORS korrekt konfiguriert
- [x] Rate-Limiting für Auth (Firebase)
- [ ] CSP-Header setzen
- [ ] XSS-Protection aktivieren

Troubleshooting

Problem: “Module not found”

Lösung: `pnpm install` im Root ausführen

Problem: “Port already in use”

Lösung:

```
pkill -f "next dev"
# oder Port ändern in package.json
```

Problem: Firebase-Connection-Fehler

Lösung:
- `config.js` vorhanden?
- Firestore-Rules deployed?
- API-Keys korrekt?

Problem: PWA funktioniert nicht

Lösung:
- HTTPS erforderlich (Production)
- Service Worker registriert?
- Manifest korrekt?

Stand: Dezember 2025