Phase 1 - Übersicht & Ergebnisse

Datum: 1. Dezember 2025
Status: ✅ Abgeschlossen

Zusammenfassung

Phase 1 des Nightlife OS Monorepo ist erfolgreich implementiert worden. Das Projekt umfasst:

- **1 Monorepo** mit Turborepo + pnpm
- **3 Shared Packages** (shared-types, core, ui)
- **6 Next.js 14 Apps** (club-app, dj-console, club-admin, staff-door, staff-waiter, staff-cloakroom)
- **Firebase-Integration** (vorbereitet)
- **i18n-Unterstützung** (DE, EN)
- **Einheitliches Theme** (Nightlife Dark Theme mit Cyan-Akzent)

Verzeichnisstruktur

```
nightlife_os/
├── packages/
│   ├── shared-types/          # TypeScript-Typen
│   │   ├── src/
│   │   │   ├── user.ts       # User-Typen (Platform + Club)
│   │   │   ├── club.ts       # Club, ClubGroup, ClubState, ClubSettings
│   │   │   ├── chat.ts       # Chat, Message
│   │   │   ├── order.ts      # Order, OrderItem
│   │   │   ├── cloakroom.ts  # CloakroomTicket
│   │   │   ├── roles.ts      # Rollen & Permissions
│   │   │   ├── api.ts        # API-Response-Typen
│   │   │   └── index.ts
│   │   ├── package.json
│   │   └── tsconfig.json
│   │
│   ├── core/                  # Firebase, Hooks, Utils
│   │   ├── src/
│   │   │   ├── firebase/
│   │   │   │   ├── init.ts          # Firebase initialisieren
│   │   │   │   ├── auth.ts          # Auth-Helpers
│   │   │   │   ├── firestore.ts     # Firestore-Wrapper
│   │   │   │   └── storage.ts       # Storage-Helpers
│   │   │   ├── hooks/
│   │   │   │   ├── use-auth.ts      # Auth-State Hook
│   │   │   │   ├── use-user-data.ts # User-Daten (Platform + Club)
│   │   │   │   ├── use-club-state.ts# Club-State Hook
│   │   │   │   ├── use-friends.ts   # Freunde-Hook
│   │   │   │   ├── use-chats.ts     # Chat-Hook
│   │   │   │   └── use-i18n.ts      # i18n-Hook
│   │   │   ├── utils/
│   │   │   │   ├── friend-code.ts   # Friend-Code-Generator
│   │   │   │   ├── trust-score.ts   # Trust-Level-Berechnung
│   │   │   │   ├── validation.ts    # Input-Validierung
│   │   │   │   └── date-time.ts     # Datums-Formatting
│   │   │   ├── constants/
│   │   │   │   ├── roles.ts         # Rollen-Definitionen
│   │   │   │   ├── permissions.ts   # Berechtigungen
│   │   │   │   └── app-config.ts    # App-Konfiguration
│   │   │   └── index.ts
│   │   ├── package.json
│   │   └── tsconfig.json
│   │
│   └── ui/                    # UI-Komponenten, i18n, Theme
│       ├── src/
│       │   ├── components/
│       │   │   ├── button.tsx           # Button-Komponente
│       │   │   ├── input.tsx            # Input-Komponente
│       │   │   ├── card.tsx             # Card-Komponente
│       │   │   ├── modal.tsx            # Modal-Komponente
│       │   │   ├── icon.tsx             # Icon-Wrapper (Lucide)
│       │   │   ├── loader.tsx           # Loading-Spinner
│       │   │   ├── toast.tsx            # Toast-Notifications
│       │   │   └── qr-code-display.tsx  # QR-Code (Platzhalter)
│       │   ├── locales/
│       │   │   ├── de.json              # Deutsche Übersetzungen
│       │   │   └── en.json              # Englische Übersetzungen
│       │   ├── theme/
│       │   │   ├── colors.ts            # Farbschema
│       │   │   ├── typography.ts        # Typografie
│       │   │   └── tailwind-preset.ts   # Tailwind-Preset
│       │   ├── utils/
│       │   │   └── cn.ts                # Tailwind-Merge-Helper
│       │   └── index.ts
│       ├── package.json
│       └── tsconfig.json
│
├── apps/
│   ├── club-app/              # 🎵 Besucher-PWA (Port 3000)
│   ├── dj-console/            # 🎛️ DJ/Lichtjockey (Port 3001)
│   ├── club-admin/            # 🏢 Club-Owner Dashboard (Port 3002)
│   ├── staff-door/            # 🚪 Türsteher-App (Port 3003)
│   ├── staff-waiter/          # 🍸 Kellner/Bar-App (Port 3004)
│   └── staff-cloakroom/       # 🧥 Garderoben-App (Port 3005)
│       └── (Jede App enthält:)
│           ├── src/
│           │   ├── app/
│           │   │   ├── layout.tsx
│           │   │   └── page.tsx
│           │   └── styles/
│           │       └── globals.css
│           ├── package.json
│           ├── next.config.js
│           ├── tailwind.config.ts
│           ├── postcss.config.js
│           └── tsconfig.json
│
├── turbo.json                 # Turborepo-Konfiguration
├── pnpm-workspace.yaml        # PNPM Workspace
├── tsconfig.json              # Basis TypeScript-Config
├── .gitignore
├── .env.example               # Beispiel-Umgebungsvariablen
├── README.md                  # Projekt-Übersicht
├── ARCHITECTURE.md            # Architektur-Dokumentation
├── FIRESTORE_SCHEMA.md        # Datenbank-Schema
└── PHASE1_OVERVIEW.md         # Diese Datei
```

Wichtige Code-Snippets

1. turbo.json (Turborepo-Konfiguration)

```json
{
  "$schema": "https://turbo.build/schema.json",
  "globalDependencies": ["**/.env.*local"],
  "pipeline": {
    "build": {
      "dependsOn": ["^build"],
      "outputs": [".next/**", "!.next/cache/**"]
    },
    "dev": {
      "cache": false,
      "persistent": true
    },
    "lint": {
      "outputs": []
    },
    "clean": {
      "cache": false
    }
  }
}
```

2. pnpm-workspace.yaml

```yaml
packages:
  - 'apps/*'
  - 'packages/*'
```

3. Root tsconfig.json

```json
{
  "compilerOptions": {
    "target": "ES2020",
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "moduleResolution": "bundler",
    "resolveJsonModule": true,
    "jsx": "preserve",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    // ...
  }
}
```

4. Firebase-Initialisierung (packages/core/src/firebase/init.ts)

```
import { initializeApp, getApps, FirebaseApp } from 'firebase/app';
import { getAuth, Auth } from 'firebase/auth';
import { getFirestore, Firestore } from 'firebase/firestore';

const firebaseConfig = {
  apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
  authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
  projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECT_ID,
  storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID,
};

export function initFirebase(): FirebaseApp {
  const apps = getApps();
  if (apps.length > 0) return apps[0];
  return initializeApp(firebaseConfig);
}

export function getFirestoreInstance(): Firestore {
  const app = initFirebase();
  return getFirestore(app);
}

export function getAuthInstance(): Auth {
  const app = initFirebase();
  return getAuth(app);
}
```

5. Auth-Hook (packages/core/src/hooks/use-auth.ts)

```
'use client';

import { useState, useEffect } from 'react';
import { User } from 'firebase/auth';
import { onAuthStateChanged } from '../firebase/auth';

export function useAuth() {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = onAuthStateChanged((user) => {
      setUser(user);
      setLoading(false);
    });
    return () => unsubscribe();
  }, []);

  return {
    user,
    loading,
    isAuthenticated: !!user,
  };
}
```

6. i18n-Hook (packages/core/src/hooks/use-i18n.ts)

```
'use client';

import { useState, useEffect, useCallback } from 'react';

const DEFAULT_LOCALE = 'de';
const SUPPORTED_LOCALES = ['de', 'en', 'fr', 'es', 'it'];

export function useI18n() {
  const [locale, setLocaleState] = useState(DEFAULT_LOCALE);
  const [translations, setTranslations] = useState({});

  useEffect(() => {
    const savedLocale = localStorage.getItem('nightlife-os-locale');
    if (savedLocale && SUPPORTED_LOCALES.includes(savedLocale)) {
      setLocaleState(savedLocale);
    }
  }, []);

  const setLocale = useCallback((newLocale: string) => {
    if (SUPPORTED_LOCALES.includes(newLocale)) {
      setLocaleState(newLocale);
      localStorage.setItem('nightlife-os-locale', newLocale);
    }
  }, []);

  const t = useCallback(
    (key: string, params?: Record<string, any>): string => {
      let translation = translations[key] || key;
      // Parameter ersetzen...
      return translation;
    },
    [translations]
  );

  return { locale, setLocale, t };
}
```

7. Button-Komponente (packages/ui/src/components/button.tsx)

```
import React from 'react';
import { cn } from '../utils/cn';

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement> {
  variant?: 'default' | 'ghost' | 'danger' | 'success';
  size?: 'sm' | 'md' | 'lg';
  fullWidth?: boolean;
}

export const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant = 'default', size = 'md', fullWidth = false, children, ...prop
s }, ref) => {
    return (
      <button
        ref={ref}
        className={cn(
          'inline-flex items-center justify-center rounded-md font-medium transition-
colors',
          {
            'bg-cyan-600 text-white hover:bg-cyan-700': variant === 'default',
            'hover:bg-slate-800 text-slate-100': variant === 'ghost',
            // ...
          },
          { 'h-10 px-4 text-base': size === 'md' },
          fullWidth && 'w-full',
          className
        )}
        {...props}
      >
        {children}
      </button>
    );
  }
);
```

8. Deutsche Lokalisierung (packages/ui/src/locales/de.json)

```json
{
  "common": {
    "welcome": "Willkommen",
    "loading": "Lade...",
    "save": "Speichern",
    "cancel": "Abbrechen"
  },
  "auth": {
    "login": "Anmelden",
    "logout": "Abmelden",
    "signup": "Registrieren",
    "email": "E-Mail",
    "password": "Passwort"
  },
  "home": {
    "title": "Startseite",
    "checkIn": "Einchecken",
    "checkOut": "Auschecken"
  }
}
```

9. Club-App Beispiel (apps/club-app/src/app/page.tsx)

```
'use client';

import { Button, Card, CardHeader, CardTitle, CardContent } from '@nightlife-os/ui';
import { useI18n } from '@nightlife-os/core';
import { Home, MessageCircle, Users, Music } from 'lucide-react';

export default function HomePage() {
  const { t, locale, setLocale } = useI18n();

  return (
    <main className="min-h-screen bg-slate-900 p-8">
      <div className="max-w-4xl mx-auto">
        <h1 className="text-4xl font-bold text-cyan-400 mb-2">
          Nightlife OS
        </h1>
        <p className="text-xl text-slate-300">
          🎵 Club App - Besucher-PWA
        </p>

        <Card className="mb-8">
          <CardHeader>
            <CardTitle>{t('common.welcome')}</CardTitle>
          </CardHeader>
          <CardContent>
            <p>Dies ist die Club-App für Gäste.</p>
          </CardContent>
        </Card>

        {/* Features */}
        <div className="grid grid-cols-2 gap-6">
          <Card hover>
            <CardHeader>
              <div className="flex items-center gap-3">
                <Home className="h-6 w-6 text-cyan-400" />
                <CardTitle>{t('home.title')}</CardTitle>
              </div>
            </CardHeader>
          </Card>
          {/* Mehr Features... */}
        </div>

        <Button variant="default" fullWidth>
          {t('auth.login')}
        </Button>
      </div>
    </main>
  );
}
```

Installation & Entwicklung

1. Dependencies installieren

```
cd /home/ubuntu/nightlife_os
pnpm install
```

Status: ✅ Erfolgreich (in 1m 28s)

2. Alle Apps gleichzeitig starten

```
pnpm dev
```

Dies startet alle Apps parallel:
- **club-app:** http://localhost:3000
- **dj-console:** http://localhost:3001
- **club-admin:** http://localhost:3002
- **staff-door:** http://localhost:3003
- **staff-waiter:** http://localhost:3004
- **staff-cloakroom:** http://localhost:3005

3. Einzelne App starten

```
# Club-App
pnpm --filter club-app dev

# DJ-Console
pnpm --filter dj-console dev

# Club-Admin
pnpm --filter club-admin dev

# Staff-Door
pnpm --filter staff-door dev

# Staff-Waiter
pnpm --filter staff-waiter dev

# Staff-Cloakroom
pnpm --filter staff-cloakroom dev
```

4. Build erstellen

```
# Alle Apps/Packages bauen
pnpm build

# Einzelnes Package bauen
pnpm --filter shared-types build
pnpm --filter core build
pnpm --filter ui build
```

Firebase-Setup

Umgebungsvariablen konfigurieren

1. Kopiere `.env.example` zu `.env`:
   `bash`
   `  cp .env.example .env`

2. Trage deine Firebase-Credentials ein:
   `env`
   `NEXT_PUBLIC_FIREBASE_API_KEY=your_api_key`
   `NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=your_project.firebaseapp.com`
   `NEXT_PUBLIC_FIREBASE_PROJECT_ID=your_project_id`
   `NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET=your_project.appspot.com`
   `NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID=your_sender_id`
   `NEXT_PUBLIC_FIREBASE_APP_ID=your_app_id`

3. Firebase-Credentials erhältst du aus der Firebase Console:
   - Gehe zu: https://console.firebase.google.com/
   - Wähle dein Projekt
   - Projekt-Einstellungen > Allgemein > Deine Apps > Web-App

---

Packages-Übersicht

shared-types

Zweck: Zentrale TypeScript-Typen
Exports:
- `PlatformUser` , `ClubUser` , `UserProfile` , `Friendship` , `FriendRequest`
- `Club` , `ClubGroup` , `ClubState` , `ClubSettings`
- `Chat` , `Message`
- `Order` , `OrderItem`
- `CloakroomTicket`
- `Role` , `RoleString` , `Permission` , `ROLE_PERMISSIONS`
- `ApiResponse` , `ApiErrorCode` , `AuthResponse` , etc.

core

Zweck: Firebase-Integration, Hooks, Utils
Exports:
- **Firebase:** `initFirebase()` , `getFirestoreInstance()` , `getAuthInstance()` , `login()` , `logout()` , etc.
- **Hooks:** `useAuth()` , `usePlatformUserData()` , `useClubUserData()` , `useClubState()` , `useFriends()` , `useChats()` , `useI18n()`
- **Utils:** `generateFriendCode()` , `validateFriendCode()` , `calculateTrustScore()` , `isValidEmail()` , `formatDate()` , etc.
- **Constants:** `ROLES` , `Permission` , `hasPermission()` , `SUPPORTED_LOCALES` , etc.

ui

Zweck: UI-Komponenten, i18n, Theme
Exports:
- **Komponenten:** `Button` , `Input` , `Card` , `Modal` , `Icon` , `Loader` , `Toast` , `QRCodeDisplay`
- **Utils:** `cn()` (Tailwind-Merge)
- **Theme:** `colors` , `typography` , `nightlifePreset`
- **Icons:** Re-Export von Lucide Icons (Home, MessageCircle, Users, Settings, etc.)

Apps-Übersicht

1. club-app (Besucher-PWA)

Port: 3000
Zielgruppe: Gäste
Features (geplant):
- Check-In/Out mit QR-Code
- Chat-System (1:1 und Gruppen)
- Freunde hinzufügen via Friend-Code
- Lichtshow/Countdown/Nachrichten-Overlays
- Gewinnspiele

2. dj-console (DJ/Lichtjockey)

Port: 3001
Zielgruppe: DJ/Lichtjockey
Features (geplant):
- Lichtshow-Steuerung (Farbe, Effekte)
- Audio-Sync (Mikrofon-Anbindung)
- Gewinnspiele starten
- Broadcast-Nachrichten senden
- Gästeliste einsehen

3. club-admin (Club-Owner Dashboard)

Port: 3002
Zielgruppe: Club-Owner/Admin
Features (geplant):
- Dashboard mit Analytics
- Personal-Verwaltung (Rollen zuweisen)
- Club-Einstellungen (Farben, Features, Öffnungszeiten)
- Abo-Verwaltung
- Multi-Club-Support

4. staff-door (Türsteher)

Port: 3003
Zielgruppe: Türsteher
Features (geplant):
- QR-Code-Scanner
- Trust-Level-Verifizierung
- Check-In/Out durchführen
- Blacklist-Prüfung
- Manueller Check-In

5. staff-waiter (Kellner/Bar)

Port: 3004
Zielgruppe: Kellner/Bar-Personal
Features (geplant):
- Bestellungen erstellen/verwalten
- Tischplan

- Bezahlung abschließen
- Bestellungs-Historie

6. staff-cloakroom (Garderobe)

Port: 3005
Zielgruppe: Garderoben-Personal
Features (geplant):
- Gegenstände einlagern/ausgeben
- Ticket-System mit QR-Code
- Ticket drucken
- Verlorene Items markieren

TypeScript-Typen Beispiele

User-Typen (shared-types/src/user.ts)

```typescript
export interface PlatformUser {
  uid: string;
  email: string;
  displayName: string | null;
  isPlatformAdmin: boolean;
  ownedClubs: string[];
  memberClubs: string[];
  createdAt: number;
  lastSeenAt: number;
}

export interface ClubUser {
  uid: string;
  email: string;
  roles: string[]; // ["guest"], ["staff", "door"], etc.
  checkedIn: boolean;
  checkedInAt: number | null;
  friendCode: string; // 7-stellig
  trustedLevel: number; // 0-100
  // ...
}
```

Club-Typen (shared-types/src/club.ts)

```
export interface Club {
  clubId: string;
  name: string;
  slug: string;
  ownerId: string;
  subscriptionTier: 'free' | 'basic' | 'pro' | 'enterprise';
  features: string[];
  theme: {
    primaryColor: string;
    secondaryColor: string;
    logo?: string;
  };
  // ...
}

export interface ClubState {
  mode: 'normal' | 'lightshow' | 'message' | 'countdown' | 'lottery_result';
  lightColor: string | null;
  lightEffect: 'color' | 'strobe' | 'psychedelic' | 'audio_sync' | null;
  countdownActive: boolean;
  // ...
}
```

Rollen (shared-types/src/roles.ts)

```
export enum Role {
  SUPER_ADMIN = 'super_admin',
  CLUB_ADMIN = 'admin',
  DJ = 'dj',
  STAFF = 'staff',
  DOOR = 'door',
  WAITER = 'waiter',
  BAR = 'bar',
  CLOAKROOM = 'cloakroom',
  GUEST = 'guest',
}

export const ROLE_PERMISSIONS: Record<RoleString, Permission[]> = {
  admin: [Permission.CLUB_SETTINGS_WRITE, ...],
  guest: [Permission.CHAT_READ_OWN, ...],
  // ...
};
```

Nächste Schritte (Phase 2)

Geplante Features:

1. **Auth-System**
   - Login/Signup mit Firebase Auth
   - User-Registration Flow
   - Protected Routes

2. **User-Management**
   - User-Profile erstellen/bearbeiten

- Friend-Code-System
- Freundschaftsanfragen

3. **Check-In/Out**
   - QR-Code-Generierung
   - QR-Code-Scanner (Türsteher)
   - Geo-Location-Verifizierung

4. **Chat-System**
   - 1:1-Chats
   - Gruppen-Chats
   - Ephemeral Images (selbstzerstörend)
   - Realtime-Updates

5. **Lichtshow-Features**
   - Farbauswahl
   - Effekte (Strobe, Psychedelic)
   - Audio-Sync (Mikrofon)
   - Countdown
   - Broadcast-Messages

Technologie-Stack Zusammenfassung

Kategorie	Technologie	Version
Framework	Next.js	14.2.28
React	React	18.2.0
Monorepo	Turborepo	Latest
Package Manager	pnpm	Latest
TypeScript	TypeScript	5.2.2
Styling	Tailwind CSS	3.3.6
Icons	Lucide React	0.294.0
Backend	Firebase	10.7.1
Utilities	date-fns	2.30.0
Utilities	clsx, tailwind-merge	Latest

Erfolgreiche Tests

✅ **pnpm install** - Erfolgreich (1m 28s)
✅ **Package-Struktur** - Alle 3 Packages erstellt
✅ **App-Scaffolding** - Alle 6 Apps erstellt
✅ **TypeScript-Typen** - Alle Typen definiert
✅ **Firebase-Integration** - Vorbereitet (Config benötigt)
✅ **i18n-System** - Basis-Setup (DE, EN)
✅ **Theme** - Nightlife Dark Theme

Bekannte Einschränkungen (Phase 1)

1. **Firebase-Config:** Keine echten Credentials (nur Platzhalter)
2. **i18n:** Nur Platzhalter-Übersetzungen (vollständige Texte in Phase 2)
3. **QR-Code:** Platzhalter-Komponente (echte QR-Integration in Phase 2)
4. **Auth-Flow:** Noch nicht implementiert (Phase 2)
5. **Realtime-Features:** Hooks vorbereitet, aber noch keine Daten (Phase 2)

Deployment-Bereitschaft

Status: ⚠️ Teilweise bereit

- ✅ Monorepo-Struktur
- ✅ Build-System (Turborepo)
- ✅ TypeScript-Konfiguration
- ✅ Next.js Apps
- ❌ Firebase-Credentials (manuell konfigurieren)
- ❌ Production-Build getestet (TODO)
- ❌ Deployment-Scripts (TODO)

Kontakt & Dokumentation

Projektinhaber: Bernhard Schirnhofer
Firma: Der Wohlstands-Code

Weitere Dokumentation:
- README.md (./README.md) - Projekt-Übersicht
- ARCHITECTURE.md (./ARCHITECTURE.md) - Vollständige Architektur
- FIRESTORE_SCHEMA.md (./FIRESTORE_SCHEMA.md) - Datenbank-Schema
- DEPLOYMENT.md (./DEPLOYMENT.md) - Deployment-Guide (TODO)

Phase 1 abgeschlossen am: 1. Dezember 2025
Nächste Phase: Phase 2 - Core Features (Auth, Chat, Check-In)