Phase 2 - Core Features (MVP) - Übersicht

Status: ✅ Implementiert
Datum: 2. Dezember 2025
Version: 2.0

🎯 Ziele von Phase 2

Phase 2 implementiert die Kern-Features für das MVP (Minimum Viable Product):

1. **Auth-System (Firebase Auth)** - Vollständiges Login/Signup/Logout
2. **User-Management & Friend-Code** - Plattform-User-Daten, Friend-Code-Generierung
3. **Check-In/Check-Out-Grundstruktur** - Basis für Club-Anwesenheit
4. **i18n-Erweiterung** - Neue Übersetzungskeys für Auth, Profile, Check-In

---

📁 Neue/Angepasste Dateien

Packages: Core

```
packages/core/src/
├── firebase/
│   └── auth.ts                    [ERWEITERT] - signUpWithEmailAndPassword, signOu-
tUser
├── hooks/
│   ├── use-auth.ts                [ERWEITERT] - signIn, signUp, signOut Methoden
│   ├── use-user-data.ts           [ERWEITERT] - usePlatformUserData, useClubUserData
│   └── use-check-in.ts            [NEU] - useCheckIn Hook
└── index.ts                       [ERWEITERT] - Export von useCheckIn
```

Packages: Shared-Types

```
packages/shared-types/src/
├── user.ts                        [ERWEITERT] - friendCode zu PlatformUser hinzuge-
fügt
├── checkin.ts                     [NEU] - CheckInRecord, CheckInStatus Typen
└── index.ts                       [ERWEITERT] - Export von checkin.ts
```

Packages: UI

```
packages/ui/src/locales/
├── de.json                        [ERWEITERT] - auth.*, profile.*, checkin.* Keys
└── en.json                        [ERWEITERT] - auth.*, profile.*, checkin.* Keys
```

Apps: Club-App

```
apps/club-app/src/app/
├── page.tsx                        [ERWEITERT] - Check-In/Out UI für eingeloggte User
└── auth/
    ├── login/
    │   └── page.tsx                [NEU] - Login-Seite
    ├── signup/
    │   └── page.tsx                [NEU] - Signup-Seite
    └── profile/
        └── page.tsx                [NEU] - Profil-Seite mit Friend-Code
```

🔑 Wichtige Code-Snippets

1. Firebase Auth (packages/core/src/firebase/auth.ts)

```typescript
/**
 * Registrierung mit E-Mail, Passwort und Display Name
 */
export async function signUpWithEmailAndPassword(
  email: string,
  password: string,
  displayName?: string
): Promise<UserCredential> {
  const auth = getAuthInstance();
  const userCredential = await createUserWithEmailAndPassword(auth, email, password);

  // Display Name setzen
  if (displayName && userCredential.user) {
    await updateProfile(userCredential.user, { displayName });

    // PlatformUser-Dokument erstellen
    await setDocument(`platform/users/${userCredential.user.uid}`, {
      uid: userCredential.user.uid,
      email: userCredential.user.email,
      displayName: displayName,
      photoURL: null,
      createdAt: Date.now(),
      lastSeenAt: Date.now(),
      isPlatformAdmin: false,
      ownedClubs: [],
      memberClubs: [],
    });
  }

  return userCredential;
}
```

2. useAuth Hook (packages/core/src/hooks/use-auth.ts)

```
export interface UseAuthReturn {
  user: User | null;
  loading: boolean;
  isAuthenticated: boolean;
  signIn: (email: string, password: string) => Promise<void>;
  signUp: (email: string, password: string, displayName?: string) => Promise<void>;
  signOut: () => Promise<void>;
}

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

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

  const signIn = useCallback(async (email: string, password: string) => {
    await signInWithEmailAndPassword(email, password);
  }, []);

  const signUp = useCallback(async (email: string, password: string, displayName?: str
ing) => {
    await signUpWithEmailAndPassword(email, password, displayName);
  }, []);

  const signOut = useCallback(async () => {
    await signOutUser();
  }, []);

  return { user, loading, isAuthenticated: !!user, signIn, signUp, signOut };
}
```

3. usePlatformUserData Hook (packages/core/src/hooks/use-user-data.ts)

```tsx
export interface UsePlatformUserDataReturn {
  user: PlatformUser | null;
  loading: boolean;
  error: Error | null;
  refreshUser: () => void;
  generateNewFriendCode: () => Promise<void>;
}

export function usePlatformUserData(uid?: string | null): UsePlatformUserDataReturn {
  const [user, setUser] = useState<PlatformUser | null>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<Error | null>(null);

  useEffect(() => {
    if (!uid) {
      setUser(null);
      setLoading(false);
      return;
    }

    // Firestore Realtime-Listener
    const unsubscribe = onDocumentSnapshot(
      `platform/users/${uid}`,
      (docSnapshot) => {
        if (docSnapshot) {
          setUser(docSnapshot as PlatformUser);
          // lastSeenAt aktualisieren
          updateDocument(`platform/users/${uid}`, {
            lastSeenAt: Date.now(),
          }).catch(console.error);
        } else {
          // Dokument existiert nicht - erstellen
          // ...
        }
        setLoading(false);
      }
    );

    return () => unsubscribe();
  }, [uid]);

  const generateNewFriendCode = async () => {
    if (!uid) return;
    const newCode = generateFriendCode();
    await updateDocument(`platform/users/${uid}`, {
      friendCode: newCode,
    });
  };

  return { user, loading, error, refreshUser, generateNewFriendCode };
}
```

4. useCheckIn Hook (packages/core/src/hooks/use-check-in.ts)

```typescript
export type CheckInStatus = 'checked_in' | 'checked_out' | 'loading';

export interface UseCheckInReturn {
  checkIn: () => Promise<void>;
  checkOut: () => Promise<void>;
  currentStatus: CheckInStatus;
  loading: boolean;
  error: Error | null;
  currentRecord: CheckInRecord | null;
}

export function useCheckIn(
  clubId: string,
  userId?: string | null
): UseCheckInReturn {
  const [currentStatus, setCurrentStatus] = useState<CheckInStatus>('loading');
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<Error | null>(null);
  const [currentRecord, setCurrentRecord] = useState<CheckInRecord | null>(null);

  // Status prüfen beim Mount
  useEffect(() => {
    // Aktuellen Check-In-Status aus Firestore laden
    // ...
  }, [clubId, userId]);

  const checkIn = useCallback(async () => {
    const recordId = `${userId}_${Date.now()}`;
    const newRecord: CheckInRecord = {
      id: recordId,
      userId,
      clubId,
      checkedInAt: Date.now(),
      checkedOutAt: null,
      via: 'manual',
    };
    await setDocument(`clubs/${clubId}/checkins/${recordId}`, newRecord);
    await updateDocument(`clubs/${clubId}/users/${userId}`, {
      checkedIn: true,
      checkedInAt: Date.now(),
    });
    setCurrentStatus('checked_in');
  }, [clubId, userId]);

  const checkOut = useCallback(async () => {
    await updateDocument(`clubs/${clubId}/checkins/${currentRecord.id}`, {
      checkedOutAt: Date.now(),
    });
    await updateDocument(`clubs/${clubId}/users/${userId}`, {
      checkedIn: false,
      checkedInAt: null,
    });
    setCurrentStatus('checked_out');
  }, [clubId, userId, currentRecord]);

  return { checkIn, checkOut, currentStatus, loading, error, currentRecord };
}
```

5. CheckInRecord Type (packages/shared-types/src/checkin.ts)

```
export interface CheckInRecord {
  id: string; // Eindeutige ID (z.B. userId_timestamp)
  userId: string; // UID des Users
  clubId: string; // ID des Clubs
  checkedInAt: number; // Unix-Timestamp (ms)
  checkedOutAt: number | null; // Unix-Timestamp (ms) oder null
  via: 'manual' | 'qr' | 'nfc' | 'auto'; // Wie wurde eingecheckt?
}

export type CheckInStatus = 'checked_in' | 'checked_out' | 'loading';
```

6. Login Page (apps/club-app/src/app/auth/login/page.tsx)

```
export default function LoginPage() {
  const router = useRouter();
  const { signIn, isAuthenticated } = useAuth();
  const { t } = useI18n();

  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);

  if (isAuthenticated) {
    router.push('/');
    return null;
  }

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    setError(null);
    setLoading(true);

    try {
      await signIn(email, password);
      router.push('/');
    } catch (err: any) {
      setError(err?.message || t('auth.loginError'));
    } finally {
      setLoading(false);
    }
  };

  return (
    <main className="min-h-screen bg-slate-900 flex items-center justify-center p-4">
      <Card>
        <form onSubmit={handleSubmit}>
          <Input type="email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
          <Input type="password" value={password} onChange={(e) => setPass-
word(e.target.value)} />
          <Button type="submit" disabled={loading}>
            {loading ? t('common.loading') : t('auth.login')}
          </Button>
        </form>
      </Card>
    </main>
  );
}
```

7. Profile Page (apps/club-app/src/app/auth/profile/page.tsx)

```
export default function ProfilePage() {
  const router = useRouter();
  const { user, isAuthenticated, signOut } = useAuth();
  const { user: platformUser, generateNewFriendCode } = usePlatformUserData(user?.uid)
;
  const { t } = useI18n();

  const [generatingCode, setGeneratingCode] = useState(false);

  if (!isAuthenticated) {
    router.push('/auth/login');
    return null;
  }

  const handleGenerateFriendCode = async () => {
    setGeneratingCode(true);
    try {
      await generateNewFriendCode();
    } finally {
      setGeneratingCode(false);
    }
  };

  return (
    <main>
      <Card>
        <p>{t('profile.displayName')}: {platformUser?.displayName}</p>
        <p>{t('auth.email')}: {user?.email}</p>

        {platformUser?.friendCode ? (
          <div>
            <p className="text-3xl font-mono">{platformUser.friendCode}</p>
          </div>
        ) : (
          <Button onClick={handleGenerateFriendCode}>
            {t('profile.generateFriendCode')}
          </Button>
        )}

        <Button onClick={signOut}>{t('auth.logout')}</Button>
      </Card>
    </main>
  );
}
```

8. Homepage mit Check-In (apps/club-app/src/app/page.tsx)

```
export default function HomePage() {
  const router = useRouter();
  const { user, loading: authLoading, isAuthenticated } = useAuth();
  const { checkIn, checkOut, currentStatus, loading: checkInLoading } = useCheckIn('de
mo-club-1', user?.uid);
  const { t } = useI18n();

  return (
    <main>
      {isAuthenticated ? (
        <Card>
          <p>{t('common.welcome')}, {user?.displayName}</p>

          <div>
            <p>Status: {currentStatus === 'checked_in'
              ? t('checkin.status.checkedIn')
              : t('checkin.status.checkedOut')}
            </p>

            {currentStatus !== 'loading' && (
              <Button
                variant={currentStatus === 'checked_in' ? 'danger' : 'success'}
                onClick={currentStatus === 'checked_in' ? checkOut : checkIn}
                disabled={checkInLoading}
              >
                {currentStatus === 'checked_in'
                  ? t('checkin.button.checkOut')
                  : t('checkin.button.checkIn')}
              </Button>
            )}
          </div>
        </Card>
      ) : (
        <Card>
          <Button onClick={() => router.push('/auth/login')}>
            {t('auth.login')}
          </Button>
          <Button onClick={() => router.push('/auth/signup')}>
            {t('auth.signup')}
          </Button>
        </Card>
      )}
    </main>
  );
}
```

🌐 i18n-Erweiterungen

Neue Keys in `de.json` und `en.json`:

```
{
  "auth": {
    "displayName": "Anzeigename" / "Display Name"
  },
  "profile": {
    "title": "Profil" / "Profile",
    "displayName": "Anzeigename" / "Display Name",
    "friendCode": "Freunde-Code" / "Friend Code",
    "generateFriendCode": "Code generieren" / "Generate Code",
    "myProfile": "Mein Profil" / "My Profile",
    "noFriendCode": "Noch kein Freunde-Code generiert" / "No friend code generated
yet"
  },
  "checkin": {
    "status": {
      "checkedIn": "Eingecheckt" / "Checked In",
      "checkedOut": "Ausgecheckt" / "Checked Out",
      "loading": "Lade Status..." / "Loading status..."
    },
    "button": {
      "checkIn": "Einchecken" / "Check In",
      "checkOut": "Auschecken" / "Check Out"
    },
    "success": {
      "checkIn": "Erfolgreich eingecheckt" / "Successfully checked in",
      "checkOut": "Erfolgreich ausgecheckt" / "Successfully checked out"
    },
    "error": {
      "checkIn": "Check-In fehlgeschlagen" / "Check-in failed",
      "checkOut": "Check-Out fehlgeschlagen" / "Check-out failed"
    }
  }
}
```

🗄️ Firestore-Schema

Neue/Genutzte Collections:

```
platform/
├── users/{uid}                    # PlatformUser mit friendCode

clubs/{clubId}/
├── users/{uid}                    # ClubUser mit checkedIn, checkedInAt
└── checkins/{checkInId}           # CheckInRecord für History
```

CheckInRecord-Struktur:

```
{
  id: "user123_1701518400000",
  userId: "user123",
  clubId: "demo-club-1",
  checkedInAt: 1701518400000,
  checkedOutAt: null,  // oder Timestamp
  via: "manual"  // oder "qr", "nfc", "auto"
}
```

✅ Features

1. Auth-System

- ✅ Email/Password Login
- ✅ Email/Password Signup mit Display Name
- ✅ Logout
- ✅ Auth-State-Management via useAuth Hook
- ✅ Automatische Erstellung von PlatformUser-Dokument
- ✅ Redirect-Logic (eingeloggt → /, nicht eingeloggt → /auth/login)

2. User-Management

- ✅ PlatformUser-Daten aus Firestore
- ✅ Realtime-Updates via onDocumentSnapshot
- ✅ Friend-Code-Generierung (7-stellig, alphanumerisch)
- ✅ Friend-Code-Anzeige auf Profil-Seite
- ✅ lastSeenAt-Tracking

3. Check-In/Check-Out

- ✅ Check-In für Demo-Club
- ✅ Check-Out mit Timestamp
- ✅ Status-Anzeige (checked_in, checked_out, loading)
- ✅ CheckInRecord-Speicherung in Firestore
- ✅ Aktualisierung von ClubUser checkedIn-Status

4. i18n

- ✅ Neue Keys für Auth, Profile, Check-In
- ✅ Deutsche und englische Übersetzungen
- ✅ Integration in alle neuen UI-Komponenten

🚀 Nächste Schritte (Phase 3)

1. **QR-Code Check-In** - QR-Code-Scanner für Türsteher
2. **Chat-System** - 1:1 und Gruppen-Chats
3. **Freundschafts-System** - Friend-Code-Eingabe, Anfragen

4. **Club-State** - Lichtshow, Countdown, Gewinnspiele
5. **PWA-Optimierung** - Service Worker, Offline-Support

📝 Hinweise zu Abweichungen

Von FIRESTORE_SCHEMA.md:

- **CheckInRecord**: Verwendet `id` statt `checkInId` als Primärschlüssel
- **via-Field**: Nutzt `'manual' | 'qr' | 'nfc' | 'auto'` statt nur `'manual' | 'qr'`
- **PlatformUser**: `friendCode` ist optional (string?) statt required

Von ursprünglicher Anforderung:

- **Alle Core-Funktionen implementiert**
- **TypeScript strict mode** aktiviert
- **Alle Texte auf Deutsch** in UI
- **Phase 1 bleibt unberührt** - keine Dateien gelöscht oder kaputt gemacht

---

Letzte Aktualisierung: 2. Dezember 2025
Version: 2.0
Status: ✅ Phase 2 abgeschlossen