Notifications & Multi-Channel Architecture

Nightlife OS - Zukunftsvision für Phase 8+

Übersicht

Phase 7 implementiert **In-App Notifications** als Foundation für ein umfassendes Multi-Channel-Notification-System. Dieses Dokument beschreibt die geplante Erweiterung für künftige Phasen.

---

1. Super-Admin-Mailing-System

Vision

- **Zielgruppe:** Alle registrierten User, oder spezifische Segmente
- **Auslöser:** Manuell durch Super-Admin oder Event-basiert
- **Integration:** Nutzt bestehende Notification-Dispatcher-Architektur

Funktionen

Segment-basiertes Targeting

```
interface MailingSegment {
  id: string;
  name: string;
  criteria: {
    clubs?: string[];              // User aus bestimmten Clubs
    roles?: PlatformRole[];        // User mit bestimmten Rollen
    language?: string[];           // User mit Sprachpräferenz
    minVisits?: number;            // Min. Besuchsanzahl
    createdAfter?: number;         // Registriert nach Datum
    lastSeenAfter?: number;        // Aktiv nach Datum
  };
}
```

Erweiterte Dispatcher-Funktion

```
/**
 * Dispatched Notifications über mehrere Kanäle
 * Phase 8+: E-Mail-Versand
 */
async function dispatchMultiChannel(
  userIds: string[],
  channels: NotificationChannel[], // ['in_app', 'email', 'push']
  title: string,
  body: string,
  data?: Record<string, any>
) {
  // In-App (bereits implementiert in Phase 7)
  if (channels.includes('in_app')) {
    await dispatchBulkNotifications(userIds, 'SYSTEM_ANNOUNCEMENT', title, body, data)
;
  }

  // E-Mail (Phase 8+)
  if (channels.includes('email')) {
    // TODO: Integration mit E-Mail-Service (z.B. SendGrid, AWS SES)
    // - Template-System für professionelle HTML-Mails
    // - KI-Übersetzung pro User.language
    // - Unsubscribe-Link
  }

  // Push (Phase 8+)
  if (channels.includes('push')) {
    // TODO: Firebase Cloud Messaging (FCM)
  }
}
```

Admin-UI (Zukunft)

- **Mailing-Editor:**
- Rich-Text-Editor für E-Mail-Body
- Vorschau für verschiedene Sprachen
- A/B-Testing-Option
- **Segment-Verwaltung:**
- Erstellung/Bearbeitung von Zielgruppen
- Vorschau der Anzahl betroffener User
- **Versand-Historie:**
- Übersicht aller gesendeten Mailings
- Öffnungsrate, Klickrate (Tracking optional)

---

2. Sprachen & KI-Übersetzung

Aktueller Stand (Phase 7)

- `PlatformUser.language` enthält bevorzugte Sprache (ISO 639-1)
- Unterstützte Sprachen: `de`, `en`, `es`, `fr`, `it`
- UI nutzt statische i18n-Files (`de.json`, `en.json`)

Zukunftsvision (Phase 8+)

Automatische Übersetzung bei Notification-Dispatch

```
/**
 * KI-gestützte Übersetzung von Notifications
 * Nutzt LLM API für natürliche Übersetzungen
 */
async function translateNotification(
  text: string,
  targetLanguage: string
): Promise<string> {
  // TODO: API-Call zu LLM Service
  // - System-Prompt: "Translate the following notification text to {targetLanguage}"
  // - Context: Club-spezifische Terminologie
  // - Fallback: Wenn LLM fehlschlägt, nutze statische Übersetzung

  const prompt = `Translate this notification to ${targetLanguage}: ${text}`;

  // Call LLM API (z.B. OpenAI, Claude, etc.)
  const translatedText = await callLLMAPI(prompt);

  return translatedText;
}
```

Workflow für Multi-Language Mailings

1. **Admin erstellt Nachricht** in Haupt-Sprache (z.B. Deutsch)
2. **System erkennt alle Ziel-User-Sprachen** aus `PlatformUser.language`
3. **KI übersetzt automatisch** in alle benötigten Sprachen
4. **Speicherung:**
   - Option A: Speichere nur Original + generiere Übersetzung on-the-fly
   - Option B: Speichere alle Übersetzungen in Firestore für Caching

Speicherstruktur (Option B)

```
interface NotificationTemplate {
  id: string;
  originalLanguage: string;
  translations: Record<string, { // ISO 639-1 -> Text
    title: string;
    body: string;
  }>;
  createdAt: number;
  createdBy: string; // Admin-UID
}

// Firestore: platform/notification_templates/{templateId}
```

---

3. Multi-Channel-Verteilung

Kanäle

Kanal	Status	Use-Case
In-App	✅ Implementiert	Echtzeit-Benachrichtigungen
Push	🔴 Geplant	Engagement außerhalb der App
E-Mail	🔴 Geplant	Wichtige Ankündigungen, Marketing
SMS	🔵 Optional	Kritische Alerts (z.B. Sicherheit)

Event-basierte Trigger

```
interface NotificationRule {
  id: string;
  event: NotificationType;
  channels: NotificationChannel[];
  priority: 'low' | 'medium' | 'high';
  userPreferences: {
    respectDoNotDisturb: boolean;
    respectEmailOptOut: boolean;
  };
}

// Beispiel: Neue Direktnachricht
const messageRule: NotificationRule = {
  id: 'rule_new_direct_message',
  event: 'NEW_DIRECT_MESSAGE',
  channels: ['in_app', 'push'],  // Kein E-Mail für einfache Messages
  priority: 'medium',
  userPreferences: {
    respectDoNotDisturb: true,
    respectEmailOptOut: true
  }
};

// Beispiel: System-Ankündigung
const systemRule: NotificationRule = {
  id: 'rule_system_announcement',
  event: 'SYSTEM_ANNOUNCEMENT',
  channels: ['in_app', 'push', 'email'], // Alle Kanäle
  priority: 'high',
  userPreferences: {
    respectDoNotDisturb: false,  // Wichtig -> ignoriere DND
    respectEmailOptOut: false    // Wichtig -> immer senden
  }
};
```

User-Einstellungen (Zukunft)

```typescript
interface NotificationPreferences {
  userId: string;
  channels: {
    in_app: { enabled: boolean };
    push: { enabled: boolean };
    email: {
      enabled: boolean;
      frequency: 'realtime' | 'daily_digest' | 'weekly_digest';
    };
  };
  doNotDisturb: {
    enabled: boolean;
    schedule?: {
      start: string; // HH:mm
      end: string;   // HH:mm
    };
  };
}

// Firestore: users/{uid}/preferences/notifications
```

---

4. Implementierungs-Roadmap

Phase 7 (Aktuell ✅)

- ✅ In-App Notifications
- ✅ Notification-Dispatcher-Architektur
- ✅ `PlatformUser.language` für Sprachpräferenz
- ✅ Broadcast-Chats (Super-Admin, Club-Admin)

Phase 8 (Geplant)

- 🔴 Firebase Cloud Messaging (FCM) Integration
- 🔴 Push-Token-Verwaltung in Firestore
- 🔴 Push-Notification bei neuen Messages/Broadcasts

Phase 9 (Geplant)

- 🔴 E-Mail-Service-Integration (SendGrid/AWS SES)
- 🔴 HTML-E-Mail-Templates
- 🔴 Super-Admin Mailing-Editor
- 🔴 Segment-basiertes Targeting

Phase 10 (Geplant)

- 🔴 KI-Übersetzung für Notifications & Mailings
- 🔴 LLM API Integration (z.B. OpenAI, Claude)
- 🔴 Translation-Cache in Firestore
- 🔴 User-Notification-Preferences (UI)

---

5. Technische Details

Firestore-Schema (Zukunft)

```
platform/
  notification_templates/{templateId}
    - originalLanguage: string
    - translations: Record<string, {title, body}>
    - createdAt: number
    - createdBy: string

users/{uid}/
  notifications/{notificationId}  (✅ Implementiert)
    - type: NotificationType
    - title: string
    - body: string
    - read: boolean
    - createdAt: number

  preferences/
    notifications  (🔴 Geplant)
      - channels: {...}
      - doNotDisturb: {...}

  push_tokens/{tokenId}  (🔴 Geplant)
    - token: string
    - platform: 'ios' | 'android' | 'web'
    - createdAt: number
```

API-Struktur (Zukunft)

```typescript
// POST /api/admin/mailings/create
interface CreateMailingRequest {
  segmentId?: string;           // Optional: Zielgruppe
  userIds?: string[];           // Optional: Direkte User-Liste
  channels: NotificationChannel[];
  title: string;
  body: string;
  translateToLanguages?: string[]; // Auto-Übersetzung
}

// POST /api/admin/mailings/send
interface SendMailingRequest {
  mailingId: string;
  scheduledAt?: number;         // Optional: Geplanter Versand
}
```

Zusammenfassung

Phase 7 legt das **Fundament** für ein umfassendes Multi-Channel-Notification-System:

1. **In-App Notifications** sind bereits implementiert und funktionsfähig
2. **Notification-Dispatcher** ist erweiterbar für zukünftige Kanäle
3. **Sprachpräferenz** (`user.language`) ist vorbereitet für KI-Übersetzung
4. **Broadcast-Chats** bieten strukturierte Admin-Kommunikation

Die nächsten Phasen ermöglichen:
- **Push Notifications** für Engagement außerhalb der App
- **E-Mail-Mailings** für Marketing und wichtige Ankündigungen
- **KI-gestützte Übersetzung** für globale Reichweite
- **User-Preferences** für personalisierte Notification-Erfahrung

Das System ist **modular und skalierbar** gestaltet, sodass jede Phase unabhängig entwickelt und getestet werden kann.