Roadmap zur Implementierung des Gamification-Features in Flutter/Firestore

Generiert von Codex

June 24, 2025

Contents

1 Einleitung

Diese Roadmap führt Schritt für Schritt durch die Implementierung des Gamification-Systems in ein bestehendes Flutter/Cloud Firestore Projekt. Sie berücksichtigt:

- 1 XP pro abgeschlossener Session an Einzelgeräten (`isMulti == false`)
- gymspezifisches Leaderboard mit opt-out Funktion
- Offline-Persistenz und Sicherstellung von max. 1 Session pro Gerät und Tag
- Multi-Gym Support (Auswahl bei mehreren Zuordnungen)
- Zukünftige Erweiterungen: Badges Level

2 1. Datenmodell in Firestore erweitern

2.1 Neue Collections Felder

a) Unter `gyms/gymId` eine Collection `leaderboard` anlegen.

- Dokument-ID: `userId`
- Felder:
  - `xp: number`
  - `updatedAt: timestamp` (optional)

b) Innerhalb jedes `leaderboard/userId` eine Subcollection `dailySessions` anlegen.

- Dokument-ID: $\texttt{deviceId}_{YYYYMMDD}Felder$:
- - `deviceId: string`
  - `date: string (YYYY-MM-DD)`

Erweiterung des Users:

- `users/userId` Feld `gymCodes: array` für Mehrfachzuordnung
- Feld `showInLeaderboard: bool` (Standard: `true`)

2.2 Beispielhafter Pfad

```
gyms/gym_01/leaderboard/DrzaD8BdlcOwumgmuYSVMj0ESAC3
xp: 5
updatedAt: 2025-06-24T10:00:00Z
dailySessions/
32c07900-3693-4faa-82fa-f41ee2d34c72_20250624
deviceId: "32c07900-3693-4faa-82fa-f41ee2d34c72"
date: "2025-06-24"
```

2.3 Dokumentation

Siehe Kapitel docs/data$_{m}odel/(internesRepo)
Offizielle Referenz: https://firebase.google.com/docs/firestore/data-model

3 2. Anpassung der Security Rules

3.1 Erweiterung für leaderboard

Füge im Block match /gyms/gymId folgenden Unterblock hinzu:

```
match /leaderboard/{userId} {
allow read: if isAdmin()
|| (request.auth != null
&& (request.auth.uid == userId
|| get(/databases/$(database)/documents/users/$(request.auth.uid)).data.gymCodes
.hasAll([gymId])));
allow create, update: if request.auth != null
&& request.auth.uid == userId
&& request.resource.data.keys().hasAny(['xp','updatedAt']);
match /dailySessions/{sessionId} {
allow create: if request.auth != null
&& request.auth.uid == userId
&& !exists(/databases/$(database)/documents/gyms/$(gymId)/leaderboard/$(userId)/dai
&& get(/databases/$(database)/documents/gyms/$(gymId)/devices/
$(request.resource.data.deviceId)).data.isMulti == false;
allow read: if request.auth != null && request.auth.uid == userId;
allow update, delete: if false;
}
}
```

3.2 Validierung weiterer Felder

Stelle sicher, dass showInLeaderboard und gymCodes in den Rules nicht überschreibbar sind.

3.3 Testing

- Mit dem Emulator (firebase emulators:start) testen
- Szenarien:
  1. XP-Eintrag auf Einzelgerät zulässig
  2. XP-Eintrag auf Multi-Gerät abgelehnt
  3. Zweiter Eintrag desselben Tages abgelehnt

4 3. Flutter Client-Logik und Offline-Support

4.1 Multi-Gym Auswahl

a) Beim Login nach Authentifizierung das Feld gymCodes des Users aus Firestore lesen.

b) Wenn mehrere gymCodes vorhanden sind:

- Navigations-Screen SelectGymScreen anzeigen, um gymId auszuwählen.
- Auswahl persistieren (z. B. in SecureStorage).

4.2 Offline-Persistenz aktivieren

In main.dart sicherstellen:

```
FirebaseFirestore.instance.settings = const Settings(
persistenceEnabled: true,
);
```

4.3 Session-Speicherung XP-Erfassung

Im Service/Provider, z.B. DeviceSessionService:

a) Beim Abschluss einer Session:

(a) Log in gyms/gymId/devices/deviceId/logs schreiben.

(b) Wenn device.isMulti == false und User showInLeaderboard == true:

i. In einer Transaction:

- Dokument dailySessions/deviceId$_{yyyyMMdd}$$erstellen(fallsnichtexistiert).Feld$xpi

5 4. Leaderboard-UI in Flutter

5.1 Darstellung

a) RankProvider erweitern:

- Methode Stream<List> watchLeaderboard(String gymId)
- Filter: nur Einträge mit showInLeaderboard == true

b) RankScreen:

- ChangeNotifierProvider verwenden.
- In initState() Provider abonnieren.
- ListView.builder für Einträge (Platz, E-Mail, XP).

- Option zum Toggle von showInLeaderboard in den Einstellungen anbieten.

c) Navigation:

- Neue Route /rank in AppRouter definieren.
- Eintrag in BottomNavigationBar oder Drawer hinzufügen.

6 5. Offline-Szenarien

- Logging + XP-Write mit Offline-Persistenz durchführen.
- Bei späterer Synchronisation:
  - Fehlschlag der XP-Write (z. B. Regelverstoß) abfangen und SnackBar("XP bereits vergeben") anzeigen.
- Im Emulator:
  1. Netzwerk trennen, Session speichern, wieder verbinden.
  2. Sicherstellen: nur ein XP-Eintrag pro Gerät/Tag.

7 6. Echtzeit-Aktualisierung

- RankProvider mit Firestore-Listener (snapshots()) implementieren.
- Automatische Aktualisierung der UI bei Datenänderungen.
- Optional: Polling-Fallback bei Problemen mit Streams.

8 7. Tests und Qualitätssicherung

8.1 Security Rules Tests

1. Tests für XP-Writes mit dem Emulator (Jest):
   - Positivfall Einzelgerät.
   - Negativfall Multi-Gerät.
   - Negativfall zweiter Write am Tag.

8.2 Flutter Widget Tests

1. Test für RankScreen:
   - Anzeige korrekter Platzierungen.
   - Toggle-Option von showInLeaderboard.

8.3 Integrationstests

- `E2E-Test mit integration`$_t est - Package$:
    - `Session abschließen, XP erhöhen, UI aktualisiert sich.`

9 8. Erweiterungen und nächste Schritte

- Multi-Gym Registrierung:
    - Nutzerprofil-Screen um Verwaltung mehrerer `gymCodes` erweitern.
    - Login Flow: Nur Screen bei `gymCodes.length > 1` anzeigen.
- Zusätzliche Gamification-Elemente:
    - Badges (Collection `gyms/gymId/badges`).
    - Level-Logik (z. B. `level = floor(sqrt(xp))`).

10 Anhang

10.1 Referenzen

- Firestore Data Modeling: `https://firebase.google.com/docs/firestore/data-model`
- Firestore Security Rules: `https://firebase.google.com/docs/rules`
- Offline Persistence: `https://firebase.flutter.dev/docs/firestore/usage/#offline-persistence`
- Flutter Integration Testing: `https://flutter.dev/docs/testing/integration-tests`