Sprint 2: NFC-Tracking & Offline-Sync – detaillierte Roadmap

Zeitraum: 18.–22. Aug 2025 (5 Arbeitstage)

Ziel: Sicherstellen, dass Trainingsdaten auch ohne Internet zuverlässig erfasst und synchronisiert werden. Die NFC-Interaktion muss robust gegen Mehrfacheingaben sein, die lokale Speicherung darf keine Daten verlieren und der Nutzer soll jederzeit über den Sync-Status informiert sein. Kleine Verbesserungen im Plan-Modus runden den Sprint ab, damit der Fokus ab Sprint 3 auf der 3D-Heatmap liegen kann.

1. Voraussetzungen und Kontext

- **Vorheriger Sprint:** Feedback-Workflow und Umfragen sind abgeschlossen; Benachrichtigungen laufen.
- **Bestehende Komponenten:** DeviceProvider speichert Sessions in Firestore, zeigt die letzte Session im UI und verwaltet XP-Zuwachs. Die aktuelle App nutzt die Firebase Offline-Persistence, aber kein dediziertes lokales Cache-System für Sessions.
- **Notwendige Libraries:** Für lokale Speicherung wird `hive` (leichtgewichtig und Flutter-friendly) oder alternativ `sqflite` eingesetzt.
- **Team:** Zwei Frontend-Entwickler, ein Backend-/Sync-Spezialist, ein QA-Engineer.

2. Aufgabenplan (Tag-für-Tag)

Tag 1 (Mo 18. Aug 2025) – Analyse & DeviceProvider-Refactoring

DeviceProvider-Code Review (AM):

- Gemeinsame Durchsicht der Methoden `loadDevice`, `addSet`, `saveWorkoutSession`.
- Identifikation von Schwachstellen:
  - Wie wird eine Session gespeichert, wenn mehrere NFC-Taps kurz hintereinander stattfinden?
  - Werden Sessions eindeutig identifiziert (z. B. über `sessionId + timestamp`)?
  - Gibt es Race Conditions beim gleichzeitigen Hinzufügen von Sets?

Design eines robusten Session-Schemas (PM):

- Neues Feld `sessionId` bei jedem `saveWorkoutSession` generieren (z. B. UUID); alle zugehörigen Sets referenzieren diese ID.
- Beim erneuten Tap prüfen, ob an diesem Tag bereits eine Session für dasselbe Gerät/Übung besteht; falls ja, die bestehende Session im Provider zur Bearbeitung laden (Multi-Tap-Prevention).
- Festlegung, wie „abgebrochene“ Sessions erkennbar und fortgesetzt werden können.

Refactoring des DeviceProvider:

- Methoden so umgestalten, dass Sets erst lokal in einer temporären Liste gespeichert werden und erst beim finalen „Speichern“ alle Sets mit derselben `sessionId` an Firestore übertragen werden.
- Fehlerbehandlung einbauen: bei Netzausfall werden Sets in ein lokalen Cache gelegt (siehe Tag 2).
- Tests vorbereiten (Mock von Firestore), um Regressionen zu vermeiden.

Tag 2 (Di 19. Aug 2025) – Lokaler Cache & Sync-Queue

Auswahl & Setup von Hive (AM):

- In `pubspec.yaml` `hive` und `hive_flutter` hinzufügen; optional `path_provider` für Speicherort.
- Eine Box `offlineSessions` eröffnen, die eine Liste von Objekten `OfflineSession` speichert (enthält `sessionId`, `deviceId`, `exerciseId`, `userId`, Liste von Sets, `timestamp`).

Implementierung der Sync-Queue (PM):

- Schritt 1: Beim Speichern einer Session im DeviceProvider prüfen, ob Internet verfügbar ist (Connectivity-Package).
- Schritt 2: Wenn offline: Session in Hive-Box mit Status `pending` ablegen.
- Schritt 3: Ein Hintergrund-Service (z. B. `sync_service.dart`) registrieren, der periodisch (oder beim Wiederherstellen der Verbindung) die Box abfragt und alle `pending`-Sessions an Firestore sendet.
- Schritt 4: Nach erfolgreichem Upload Session aus der Box entfernen oder auf `synced` setzen.

Edge-Cases berücksichtigen:

- Duplicate-Uploads verhindern (per `sessionId`).
- Konflikt zwischen lokal gespeicherten Sessions und währenddessen online entstandenen Sessions lösen (z. B. Merge oder Benutzerabfrage).

Tests & Mocks:

- Unit-Tests für `sync_service`: Simulieren von Offline-/Online-Zustand; Prüfen, dass `pending`-Sessions korrekt übertragen werden; Verhindern von Doppel-Uploads.
- QA-Engineer erstellt Test-Daten und nutzt Firebase Emulator, um Uploads zu beobachten.

Tag 3 (Mi 20. Aug 2025) – UI-Feedback & Nutzer-Erfahrung

Design-Entwurf für Sync-Status-Icon (AM):

- In Zusammenarbeit mit UI-/UX-Designer Icon-Set definieren:
  - `onlineSync` (grünes Wolken-Icon mit Haken) → Alle Daten synchronisiert.
  - `offlinePending` (rotes/gelbes Wolken-Icon) → Es gibt ausstehende Uploads.
  - `syncing` (animiertes Loader-Icon) → Upload im Fortschritt.

Implementierung im Frontend (PM):

- Eine `SyncStatusProvider` Klasse erstellen, die den Status (online, offline, syncing, error) hält und von Widgets beobachtet werden kann.
- Die `HomeScreen`-AppBar und ggf. `DeviceScreen` erweitern, um das Status-Icon rechts oben anzuzeigen.
- Tooltips hinzufügen („X ausstehende Sessions zum Synchronisieren“).
- Zustandswechsel triggert einen kurzen Snackbar-Hinweis („Daten werden synchronisiert…“) oder offline-Banner.

Benutzeroptionen:

- Optionales Menü „Jetzt synchronisieren“ im Status-Icon, das den Sync-Service manuell triggert.
- Einstellung im Profil („Nur über WLAN synchronisieren“).

Tag 4 (Do 21. Aug 2025) – End-to-End-Tests & Plan-Modus-Verbesserungen

Integrationstest „NFC-Scan → Offline → Online-Sync“ (AM):

- Mit `integration_test`-Package einen Test schreiben, der folgende Schritte durchführt:
  - App starten, authentifizieren, Gym auswählen.
  - Netzverbindung trennen (mithilfe von `connectivity_plus` Mocks oder Android/iOS Simulators).
  - `DeviceScreen` via NFC-Simulator öffnen, einige Sets eingeben und speichern.
  - Prüfen, dass die Session im lokalen Cache vorhanden ist und UI den Offline-Status anzeigt.
  - Netzverbindung wiederherstellen → `SyncStatusProvider` wechselt auf syncing und anschließend online; Session existiert in Firestore.
  - UI zeigt „Letzte Session: . . .“ korrekt an.

Plan-Modus-Verbesserungen (PM):

- Resttimer anpassbar: Im Plan-Editor optionales Feld „Pause (Sekunden)“ pro Übung; Default 90 s; Implementierung in `RestTimerWidget`.
- RIR (Reps in Reserve) optional: Das Eingabefeld für RIR im `DeviceScreen` nur auf Wunsch anzeigen; konfigurierbar in den Plan-Einstellungen oder global via Settings.
- Multi-Tap-Prevention: NFC-Listener so anpassen, dass ein erneuter Tap innerhalb von 5 Sekunden ignoriert wird oder die laufende Session wieder geöffnet wird; Feedback per Toast („Session läuft bereits“).
- QA-Tests für diese Änderungen (Widget-Tests und Integrationstests).

Tag 5 (Fr 22. Aug 2025) – Finalisierung & Review

Code-Review & Refactoring:

- Gemeinsame Durchsicht des neuen Caching-Mechanismus, Sync-Services und UI-Elemente; Abgleich mit Coding-Guidelines.
- Überprüfung der `sessionId`-Logik; Sicherstellen, dass Firestore-Dokumente konsistent sind.
- Cleanup von Alt-Code; Entfernen eventuell redundanter Felder.

Test-Abschluss und CI-Validierung:

- Alle neu geschriebenen Unit-Tests, Widget-Tests und Integrationstests laufen lokal und in GitHub Actions grün.
- Code-Coverage überprüfen (idealerweise gesteigert).
- QA-Engineer erstellt einen Test-Report mit gefundenen Bugs und offenen Punkten.

Dokumentation ergänzen:

- `docs/data_model.md`: Beschreibung des neuen Session-Schemas (inkl. `sessionId`, pendingSessions in Hive).
- `docs/offline_sync.md` (neu): Erläutert Architektur des Offline-Caches, Sync-Queue, Icon-Status und Benutzeroptionen.

- Plan-Modus-Dokumentation: Beschreibung der neuen Resttimer-Option und optionalen RIR-Felder.

Sprint-Review & Übergabe in Sprint 3:

- Demo: „NFC-Scan → offline arbeiten → später synchronisieren“.
- Kurzer Ausblick auf Sprint 3 (3D-Heatmap & Gamification-Erweiterungen) mit Klarstellung, dass das Logging/Sync nun stabil als Grundlage dient.
- Backlog-Items für Sprint 3 definieren (Feintuning, eventuell Priorisierung zusätzlicher Sync-Features wie „Delta-Sync“ oder „Konfliktauflösung bei parallelem Bearbeiten“).

3. Definition of Done

- Robuster `DeviceProvider` mit eindeutigen `sessionIds` und Multi-Tap-Prevention.
- Lokaler Offline-Cache via Hive, der Sessions speichert, wenn keine Internetverbindung besteht, und sie automatisch bei Wiederverbindung synchronisiert.
- Sync-Status-UI mit Icons in der App-Leiste, die den Synchronisationszustand klar anzeigen und optional manuell ausgelöst werden können.
- End-to-End-Tests demonstrieren erfolgreiches Logging, Offline-Nutzung und anschließenden Sync; Plan-Modus-Verbesserungen funktionieren.
- Dokumentation aktualisiert und beschreibt das neue Sync-Verhalten und Session-Modell.
- Alle Tests bestehen in der CI; Code wurde reviewed und gemerged.

4. Ausblick auf Sprint 3

Mit dem stabilen Offline-Sync und der verbesserten NFC-Interaktion kann das Team in Sprint 3 (25.–29. Aug 2025) die 3D-Heatmap und erweiterte Gamification-Features angehen, ohne sich um Datenverlust oder Synchronisationsprobleme sorgen zu müssen.