# Strategischer Plan zur Entwicklung einer plattformübergreifenden Fitness-Anwendung für Android und Windows

## I. Zusammenfassung und strategischer Überblick

Dieses Dokument stellt einen umfassenden strategischen und technischen Plan für die Konzeption, Entwicklung und Bereitstellung einer plattformübergreifenden Fitnessanwendung dar. Die Anwendung ist für den Einsatz auf Android-Mobilgeräten und Windows-PCs konzipiert und erfüllt die spezifischen Anforderungen an die Erstellung und Planung von Trainingseinheiten, die Nachverfolgung von Übungen und Gewichten sowie die nahtlose Datensynchronisation zwischen den Plattformen. Das Kernziel ist die Bereitstellung einer robusten, performanten und benutzerfreundlichen Lösung, die den Anwender bei der Erreichung seiner Fitnessziele unterstützt.

Die Analyse der Projektanforderungen und der aktuellen Technologielandschaft führt zu den folgenden zentralen strategischen Empfehlungen, die die Grundlage für diesen Plan bilden:

* **Empfehlung für das Frontend-Framework:** Die Entwicklung der Client-Anwendungen für Android und Windows sollte mit **.NET MAUI** (Multi-platform App UI) erfolgen. Dieses von Microsoft entwickelte Framework ermöglicht die Erstellung nativer Anwendungen für beide Zielplattformen aus einer einzigen C#-Codebasis. Die Wahl von.NET MAUI ist strategisch begründet, da es eine erstklassige Unterstützung für Windows als primäre Zielplattform bietet und gleichzeitig eine hohe Performance und ein natives Benutzererlebnis auf Android gewährleistet. Dies minimiert den Entwicklungsaufwand und stellt eine konsistente Funktionalität sicher.
* **Empfehlung für die Backend-Plattform:** Als Backend- und Datensynchronisations-Engine wird **Supabase** empfohlen. Supabase ist eine Open-Source-Alternative zu Firebase, die auf einer robusten PostgreSQL-Datenbank aufbaut. Diese Wahl ist durch die stark relationale Natur von Fitnessdaten begründet (Benutzer, Trainingspläne, Übungen, Sätze). Das relationale Modell von PostgreSQL ist für diese Art von Datenstruktur fundamental besser geeignet als NoSQL-Alternativen. Zudem bietet Supabase leistungsstarke, integrierte Funktionen wie Echtzeit-Datenabonnements und Row Level Security (RLS), die für die sichere und sofortige Synchronisation der Benutzerdaten zwischen den Geräten unerlässlich sind.

Das prognostizierte Ergebnis dieser technologischen Ausrichtung ist eine hochperformante, sichere und skalierbare Anwendung, die alle funktionalen Anforderungen erfüllt. Der vorgeschlagene Technologie-Stack basiert auf modernen, wartbaren und kosteneffizienten Grundlagen, die eine solide Basis für zukünftige Erweiterungen schaffen. Die folgenden Abschnitte dieses Dokuments werden diese Empfehlungen detailliert analysieren, begründen und einen konkreten Umsetzungsplan für die Architektur, die Kernfunktionen, das User Experience Design sowie die Entwicklungs- und Bereitstellungsphasen vorlegen.

## II. Fundamentale Architektur: Auswahl des Technologie-Stacks

Die Auswahl des richtigen Technologie-Stacks ist die wichtigste strategische Entscheidung, die den gesamten Lebenszyklus der Anwendung beeinflusst. Sie bestimmt die Entwicklungsgeschwindigkeit, die Performance, die Skalierbarkeit und die Wartbarkeit des Endprodukts. In diesem Abschnitt werden die Optionen für das Frontend-Framework und die Backend-Plattform rigoros analysiert, um eine datengestützte und fundierte Empfehlung zu geben, die optimal auf die Projektanforderungen zugeschnitten ist.

### 2.1. Empfehlung für das plattformübergreifende Frontend-Framework

Die zentrale Herausforderung bei der Entwicklung der Client-Anwendung liegt in der Auswahl eines Frameworks, das sowohl auf Android als auch auf dem Windows-Desktop eine erstklassige Leistung und ein natives Benutzererlebnis bietet. Während der Markt für mobile plattformübergreifende Entwicklung von bestimmten Technologien dominiert wird, erfordert die explizite Anforderung einer hochwertigen Windows-PC-Anwendung eine differenzierte Betrachtung.

#### Vergleichende Analyse der Frameworks

Eine Analyse der führenden plattformübergreifenden Frameworks im Jahr 2025 zeigt ein klares Bild der Stärken und Schwächen der jeweiligen Optionen.1

* **.NET MAUI:** Als Weiterentwicklung von Xamarin.Forms ist.NET MAUI tief in das Microsoft-Ökosystem integriert und verwendet C# sowie die.NET-Plattform.3 Es ist speziell dafür konzipiert, native Anwendungen für Windows, Android, iOS und macOS aus einem einzigen Projekt und einer einzigen Codebasis zu erstellen.3 Die besondere Stärke von.NET MAUI liegt in der Behandlung von Windows als primäre Zielplattform, bei der die moderne Windows UI 3 (WinUI 3) Bibliothek für die Darstellung der Benutzeroberfläche genutzt wird. Dies gewährleistet eine nahtlose Integration in das Windows-Betriebssystem und den Zugriff auf native APIs, was zu einer authentischen und performanten Desktop-Anwendung führt.5 Die enge Integration mit Visual Studio bietet zudem eine hochproduktive Entwicklungsumgebung.3
* **Flutter:** Entwickelt von Google, verwendet Flutter die Programmiersprache Dart und ist bekannt für seine Fähigkeit, visuell ansprechende und hochperformante Benutzeroberflächen zu erstellen.2 Flutter erreicht eine nahezu native Performance durch die direkte Kompilierung in Maschinencode und verfügt über eine umfangreiche Bibliothek an anpassbaren Widgets.2 Obwohl Flutter die Entwicklung von Windows-Anwendungen unterstützt, liegt der Schwerpunkt des Frameworks, der Community und des Ökosystems eindeutig auf mobilen Plattformen (Android und iOS). Die Windows-Unterstützung ist zwar funktional, erreicht aber nicht die gleiche Reife und native Integration wie bei.NET MAUI.
* **React Native:** Entwickelt von Meta, ermöglicht React Native die Entwicklung von Anwendungen mit JavaScript/TypeScript und dem React-Paradigma.2 Es verfügt über eine der größten Entwicklergemeinschaften und ein riesiges Ökosystem an Bibliotheken.2 Ähnlich wie bei Flutter ist die primäre Stärke von React Native die mobile Entwicklung. Die Unterstützung für Windows ist vorhanden, kann aber in Bezug auf die Verfügbarkeit von nativen Modulen und die nahtlose Integration hinter der mobilen Entwicklung zurückbleiben.8
* **Kotlin Multiplatform (KMP):** Dieser Ansatz ermöglicht es, die Geschäftslogik in Kotlin zu schreiben und über verschiedene Plattformen hinweg zu teilen, während die Benutzeroberfläche (UI) für jede Plattform nativ entwickelt wird (z. B. mit Jetpack Compose für Android und Swift/Objective-C für iOS).8 Für die Windows-Anwendung müsste ebenfalls eine separate, native UI erstellt werden. Dieser Ansatz bietet die höchste UI-Fidelität und Performance, erhöht aber den Entwicklungsaufwand erheblich, da effektiv mehrere UIs gewartet werden müssen, was dem Ziel einer einzigen Codebasis für die gesamte Anwendung widerspricht.1

#### Begründung der Empfehlung

**Die strategische Empfehlung lautet.NET MAUI.**

Die entscheidende Variable in dieser Analyse ist die explizite Anforderung einer vollwertigen und erstklassigen Anwendung für den **Windows PC**. Während Flutter und React Native exzellente Wahlmöglichkeiten für rein mobile Projekte sind, behandeln sie den Desktop oft als sekundäre Plattform..NET MAUI hingegen ist von seinem Hersteller Microsoft als primäres Werkzeug für die moderne Windows-Entwicklung positioniert, das gleichzeitig eine hervorragende Unterstützung für Android bietet. Diese Ausrichtung minimiert das Risiko, dass die Windows-Anwendung wie eine schlecht angepasste mobile App wirkt. Stattdessen wird sie sich wie eine native Desktop-Anwendung anfühlen und verhalten, was für die Akzeptanz bei PC-Nutzern entscheidend ist. Die Verwendung einer einzigen Sprache (C#) und eines einzigen Frameworks (.NET) für die gesamte Anwendungslogik und UI auf beiden Plattformen stellt die effizienteste und wartbarste Lösung für die gestellten Anforderungen dar.

| Kriterium | .NET MAUI | Flutter | React Native | Kotlin Multiplatform (KMP) |
| --- | --- | --- | --- | --- |
| **Primäre Sprache** | C# | Dart | JavaScript / TypeScript | Kotlin |
| **Windows-Unterstützung** | Erstklassig (Nativ via WinUI 3) | Gut (Nicht-nativ gerendert) | Gut (Abhängig von Community-Modulen) | Nur Geschäftslogik; UI muss separat entwickelt werden |
| **Android-Unterstützung** | Erstklassig (Nativ) | Erstklassig (Nativ gerendert) | Erstklassig (Nativ) | Erstklassig (Nativ) |
| **Performance** | Nativ | Nahezu nativ | Nahezu nativ | Nativ |
| **Ökosystem/Community** | Stark (.NET-Ökosystem) | Sehr groß | Riesig | Wachsend |
| **Entwicklungsaufwand** | Gering (einheitliche Codebasis) | Gering (einheitliche Codebasis) | Gering (einheitliche Codebasis) | Hoch (separate UIs) |
| **Strategische Eignung** | **Optimal** | Gut | Gut | Mäßig |

### 2.2. Backend und Datensynchronisations-Engine

Das Backend dient als zentraler Hub für Benutzerdaten, Authentifizierung und die Echtzeit-Synchronisation zwischen dem Android-Gerät und dem Windows-PC. Ein Backend-as-a-Service (BaaS) ist hier die effizienteste Lösung, da es die Komplexität der Serververwaltung abstrahiert und vorgefertigte Lösungen für gängige Anforderungen bietet.9 Die beiden führenden Anbieter in diesem Bereich sind Googles Firebase und das Open-Source-Projekt Supabase.11

#### Vergleichende Analyse: Supabase (SQL) vs. Firebase (NoSQL)

Die Wahl zwischen diesen beiden Plattformen ist fundamental eine Entscheidung über das zugrunde liegende Datenmodell, das weitreichende Konsequenzen für die Architektur, Sicherheit und Skalierbarkeit der Anwendung hat.

* **Datenmodell und Abfrageflexibilität:** Die Daten einer Fitnessanwendung sind von Natur aus stark relational. Ein Benutzer hat mehrere Trainingspläne. Ein Trainingsplan besteht aus mehreren Übungen. Bei jeder Trainingseinheit werden mehrere Sätze für eine Übung protokolliert. Diese hierarchischen und vernetzten Beziehungen lassen sich ideal in einem relationalen SQL-Datenbankmodell abbilden, wie es **Supabase** mit **PostgreSQL** bietet.11 Mit SQL können komplexe Abfragen, Joins und Transaktionen einfach und effizient durchgeführt werden, um beispielsweise den Fortschritt eines Benutzers bei einer bestimmten Übung über Monate hinweg zu analysieren.15  
    
  Firebase hingegen verwendet eine NoSQL-Dokumentendatenbank (Firestore).13 Das Speichern von relationalen Daten in einer solchen Struktur erfordert entweder eine Denormalisierung (das Duplizieren von Daten über verschiedene Dokumente hinweg) oder komplexe, mehrstufige Abfragen auf der Client-Seite, um die Beziehungen aufzulösen.13 Dies kann zu Dateninkonsistenzen, erhöhtem Speicherbedarf und einer komplexeren und fehleranfälligeren Anwendungslogik führen, insbesondere wenn die Komplexität der Datenbeziehungen wächst.
* **Sicherheit:** **Supabase** integriert die **Row Level Security (RLS)** von PostgreSQL nahtlos mit seinem Authentifizierungssystem.18 RLS ermöglicht die Definition von feingranularen Zugriffsregeln direkt auf Datenbankebene mittels SQL. Eine einfache Regel wie  
  auth.uid() = user\_id stellt sicher, dass ein Benutzer *ausschließlich* auf seine eigenen Daten zugreifen kann, unabhängig von der Client-Anwendung.19 Diese serverseitige, datenbanknahe Sicherheitsdurchsetzung ist extrem robust.  
    
  Firebase verwendet ein JSON-basiertes Regelsystem, das ebenfalls sehr leistungsfähig ist, aber bei komplexen relationalen Zugriffsmustern (z.B. "erlaube Zugriff auf Dokument X nur, wenn der Benutzer auch Besitzer von Dokument Y in einer anderen Sammlung ist") schnell unübersichtlich werden kann.
* **Open Source und Portabilität:** **Supabase** ist ein Open-Source-Projekt, das auf etablierten offenen Standards wie PostgreSQL aufbaut.11 Dies bietet den strategischen Vorteil, dass keine Abhängigkeit von einem einzigen Anbieter (Vendor Lock-in) entsteht. Im Bedarfsfall könnte die gesamte Plattform selbst gehostet werden, was maximale Kontrolle und Flexibilität gewährleistet.17  
    
  Firebase ist ein proprietäres Produkt von Google, was eine starke Abhängigkeit vom Google-Ökosystem bedeutet.13
* **Echtzeit- und Offline-Fähigkeiten:** Beide Plattformen bieten leistungsstarke Echtzeit-Funktionen, um Datenänderungen sofort an alle verbundenen Clients zu senden.15 Firebase verfügt über einen ausgereiften, integrierten Offline-Caching-Mechanismus, der für einfache Anwendungsfälle gut funktioniert.21 Supabase erfordert für eine echte Offline-First-Architektur die Integration von Drittanbieter-Bibliotheken oder eine benutzerdefinierte Logik, was zunächst mehr Aufwand bedeutet, aber letztendlich zu einer flexibleren und robusteren Lösung führt, wie in Abschnitt 3.2 detailliert beschrieben wird.22

#### Begründung der Empfehlung

**Die strategische Empfehlung lautet Supabase.**

Die Wahl des Backends muss von der Struktur der zu verwaltenden Daten geleitet werden. Die relationale Natur der Fitnessdaten macht eine SQL-Datenbank zur architektonisch überlegenen Wahl. Die Entscheidung für Firebase (NoSQL) würde einen architektonischen Kompromiss darstellen, der die Anwendungslogik zwingt, die Unzulänglichkeiten der Datenbank auszugleichen, anstatt deren Stärken zu nutzen. Supabase mit PostgreSQL ist das richtige Werkzeug für diese Aufgabe. Die Kombination aus der Abfragemächtigkeit von SQL, der robusten Sicherheit durch RLS und der strategischen Freiheit durch den Open-Source-Ansatz macht Supabase zur idealen Grundlage für eine langlebige und skalierbare Fitnessanwendung.

| Kriterium | Supabase | Firebase |
| --- | --- | --- |
| **Datenbankmodell** | Relational (PostgreSQL) | NoSQL (Firestore) |
| **Eignung für Fitnessdaten** | **Optimal** (stark relational) | Mäßig (erfordert Denormalisierung) |
| **Abfragesprache** | SQL (mächtig, standardisiert) | Proprietäre Abfragen (eingeschränkt) |
| **Sicherheitsmodell** | Row Level Security (RLS) in der DB | JSON-basierte Regeln |
| **Portabilität** | Hoch (Open Source, self-hostable) | Gering (Proprietär, Vendor Lock-in) |
| **Echtzeit-Synchronisation** | Ja | Ja |
| **Offline-Unterstützung** | Konfigurierbar (flexibel) | Integriert (einfach, aber limitiert) |

### 2.3. Die empfohlene Full-Stack-Architektur

Basierend auf den vorangegangenen Analysen ergibt sich eine klare und kohärente Full-Stack-Architektur. Diese Architektur maximiert die Synergien zwischen den ausgewählten Technologien, um eine effiziente Entwicklung und ein performantes Endprodukt zu gewährleisten.

Die Architektur besteht aus drei Hauptkomponenten:

1. **Client-Tier (.NET MAUI):** Dies ist die Anwendung, die auf den Geräten der Benutzer (Android und Windows PC) läuft. Sie wird als einzelnes C#-Projekt in Visual Studio entwickelt. Diese Schicht ist verantwortlich für:
   * Die Darstellung der Benutzeroberfläche (UI), die in XAML (einer deklarativen Markup-Sprache) definiert wird.
   * Die Implementierung der Anwendungs- und Präsentationslogik in ViewModels, die dem MVVM-Muster folgen.
   * Die lokale Datenpersistenz in einer verschlüsselten SQLite-Datenbank, um eine vollständige Offline-Funktionalität zu gewährleisten.
2. **Kommunikationsschicht (supabase-csharp):** Die gesamte Kommunikation zwischen der.NET MAUI-Anwendung und dem Supabase-Backend wird über die offizielle supabase-csharp-Client-Bibliothek abgewickelt.25 Diese Bibliothek kapselt die Komplexität der API-Aufrufe und bietet eine typsichere Schnittstelle für:
   * Benutzerauthentifizierung (Anmeldung, Registrierung, Sitzungsverwaltung).26
   * Datenbankoperationen (CRUD) über die von Supabase automatisch generierte PostgREST-API.27
   * Das Abonnieren von Echtzeit-Datenänderungen aus der Datenbank.29
3. **Backend-Tier (Supabase):** Dies ist die in der Cloud gehostete Plattform, die alle serverseitigen Dienste bereitstellt.14
   * **PostgreSQL-Datenbank:** Das Herzstück des Systems, das alle Anwendungsdaten speichert.
   * **Supabase Auth:** Der Dienst, der die Benutzeridentitäten verwaltet und sichere JWTs (JSON Web Tokens) für die API-Kommunikation ausstellt.
   * **Realtime Engine:** Der Dienst, der Datenbankänderungen überwacht und diese über WebSockets an die abonnierten Clients sendet.
   * **Row Level Security (RLS):** Die in der Datenbankebene durchgesetzten Sicherheitsrichtlinien, die den Datenzugriff für jeden Benutzer individuell regeln.18

Diese Architektur schafft eine klare Trennung der Verantwortlichkeiten. Die Client-Anwendung konzentriert sich auf die Benutzererfahrung und die Offline-Fähigkeit, während das Supabase-Backend die Datenhaltung, Sicherheit und Echtzeit-Synchronisation zuverlässig und skalierbar übernimmt.

## III. Anwendungs-Blueprint: Implementierung der Kernfunktionen

Nach der Festlegung der fundamentalen Architektur beschreibt dieser Abschnitt den konkreten Plan zur Umsetzung der Kernfunktionen der Anwendung. Dies umfasst die Datenmodellierung, die Strategie für die Offline-Synchronisation, die Benutzerauthentifizierung und die detaillierte Ausgestaltung des Workout-Lebenszyklus.

### 3.1. Datenmodellierung und Persistenzstrategie

Ein durchdachtes Datenmodell ist die Grundlage für eine funktionale und performante Anwendung. Die Datenstruktur muss die Beziehungen zwischen den verschiedenen Entitäten der Fitness-Domäne logisch und effizient abbilden.

#### Cloud-Datenbankschema (PostgreSQL in Supabase)

Das folgende Schema wird für die zentrale PostgreSQL-Datenbank in Supabase vorgeschlagen. Es orientiert sich an bewährten Praktiken für die Modellierung von Fitnessdaten und ist für Skalierbarkeit und Abfrageeffizienz optimiert.31

* users:
  + id (UUID, Primary Key, referenziert auth.users.id): Eindeutige Benutzer-ID.
  + username (TEXT): Anzeigename des Benutzers.
  + goal (TEXT): Hauptziel des Benutzers (z.B. "Kraftaufbau", "Gewichtsverlust").
  + experience\_level (TEXT): Erfahrungslevel (z.B. "Anfänger", "Fortgeschritten").
  + preferred\_unit (TEXT): Bevorzugte Gewichtseinheit ('kg' oder 'lbs').
* exercises:
  + id (SERIAL, Primary Key): Eindeutige ID für eine Übung.
  + name (TEXT, Unique): Name der Übung (z.B. "Bankdrücken", "Kniebeugen").
  + muscle\_group (TEXT): Hauptmuskelgruppe (z.B. "Brust", "Beine").
  + description (TEXT): Optionale Beschreibung oder Anleitung.
* workout\_plans:
  + id (UUID, Primary Key): Eindeutige ID für einen Trainingsplan.
  + user\_id (UUID, Foreign Key zu users.id): Der Benutzer, dem der Plan gehört.
  + name (TEXT): Name des Plans (z.B. "Push-Tag A", "Ganzkörpertraining").
  + created\_at (TIMESTAMPZ): Zeitstempel der Erstellung.
* plan\_exercises:
  + plan\_id (UUID, Foreign Key zu workout\_plans.id): Verweis auf den Trainingsplan.
  + exercise\_id (INTEGER, Foreign Key zu exercises.id): Verweis auf die Übung.
  + display\_order (INTEGER): Reihenfolge der Übung im Plan.
  + (Composite Primary Key auf plan\_id, exercise\_id)
* workout\_sessions:
  + id (UUID, Primary Key): Eindeutige ID für eine absolvierte Trainingseinheit.
  + user\_id (UUID, Foreign Key zu users.id): Der Benutzer, der das Training absolviert hat.
  + plan\_name\_snapshot (TEXT): Name des Plans zum Zeitpunkt des Trainings.
  + started\_at (TIMESTAMPZ): Beginn des Trainings.
  + completed\_at (TIMESTAMPZ): Ende des Trainings.
* set\_logs:
  + id (SERIAL, Primary Key): Eindeutige ID für einen protokollierten Satz.
  + session\_id (UUID, Foreign Key zu workout\_sessions.id): Verweis auf die Trainingseinheit.
  + exercise\_id (INTEGER, Foreign Key zu exercises.id): Verweis auf die durchgeführte Übung.
  + set\_number (INTEGER): Die Nummer des Satzes innerhalb der Übung für diese Einheit.
  + weight (DECIMAL): Das verwendete Gewicht.
  + reps (INTEGER): Die Anzahl der Wiederholungen.
  + rest\_time\_seconds (INTEGER): Die nach diesem Satz eingelegte Pause in Sekunden.
  + logged\_at (TIMESTAMPZ): Zeitstempel der Protokollierung.

#### Lokale Datenbank (SQLite) und Datensicherheit

Die Client-Anwendung wird eine lokale SQLite-Datenbank verwenden, deren Schema die Struktur der Cloud-Datenbank exakt widerspiegelt. Für die Interaktion mit der Datenbank in.NET MAUI wird die bewährte Bibliothek sqlite-net-pcl eingesetzt, die eine einfache objektrelationale Abbildung (ORM) ermöglicht.34

Ein entscheidender Aspekt ist die Sicherheit der lokal gespeicherten Daten. Da es sich um sensible Gesundheitsdaten handelt, ist eine Verschlüsselung zwingend erforderlich, um die Privatsphäre der Nutzer zu schützen und regulatorischen Anforderungen wie der DSGVO zu entsprechen.37 Zu diesem Zweck wird

**SQLCipher** in die Anwendung integriert. SQLCipher ist eine Erweiterung für SQLite, die eine transparente 256-Bit-AES-Verschlüsselung der gesamten Datenbankdatei ermöglicht.38 Die Implementierung erfolgt durch die Verwendung der SQLCipher-kompatiblen NuGet-Pakete, die eine sichere Speicherung der Daten auf dem Gerät gewährleisten, ohne die Anwendungslogik wesentlich zu verändern.39

### 3.2. Design für eine Offline-First-Erfahrung

Eine Fitness-App muss in Umgebungen mit unzuverlässiger oder fehlender Internetverbindung, wie sie in vielen Fitnessstudios vorkommt, einwandfrei funktionieren.21 Eine reine Caching-Strategie, wie sie von einigen Plattformen standardmäßig angeboten wird, ist hierfür oft unzureichend.21 Stattdessen ist eine "Offline-First"-Architektur erforderlich, bei der die Anwendung so konzipiert ist, dass sie primär mit der lokalen Datenbank interagiert. Die Synchronisation mit der Cloud wird zu einem Hintergrundprozess, der die Benutzererfahrung nicht blockiert.

#### Vorgeschlagene Synchronisationslogik

Die Implementierung folgt dem "Outbox Pattern", einer robusten und bewährten Methode zur Handhabung von Offline-Datenänderungen.

1. **Leseoperationen (Reads):** Alle Anfragen der Benutzeroberfläche zum Abrufen von Daten (z.B. das Laden eines Trainingsplans oder der bisherigen Leistungen) werden **immer und ausschließlich** von der lokalen, verschlüsselten SQLite-Datenbank bedient. Dies garantiert eine sofortige Reaktion der UI, unabhängig vom Netzwerkstatus, und eliminiert Latenz.
2. **Schreiboperationen (Writes):** Wenn ein Benutzer eine datenverändernde Aktion durchführt (z.B. einen Satz protokolliert, einen neuen Plan erstellt oder eine Übung löscht), wird diese Änderung **sofort in die lokale SQLite-Datenbank geschrieben**. Zusätzlich wird in der entsprechenden Tabelle ein Statusfeld (z.B. is\_synced = false) gesetzt, um die Änderung als noch nicht mit dem Server synchronisiert zu markieren. Die Benutzeroberfläche wird sofort aktualisiert, um dem Benutzer optimistisches Feedback zu geben, dass seine Aktion erfolgreich war.
3. **Echtzeit-Synchronisation (bei Online-Verbindung):** Sobald die Anwendung eine stabile Internetverbindung hat, baut sie eine Echtzeit-Verbindung zu Supabase auf und abonniert Änderungen an den Daten des angemeldeten Benutzers.30 Eingehende Änderungen von anderen Geräten (z.B. Daten, die auf dem Windows-PC eingegeben wurden) werden empfangen und direkt in die lokale SQLite-Datenbank geschrieben. Dies hält die lokalen Daten auf dem neuesten Stand.
4. **Hintergrund-Synchronisationsdienst:** Ein separater, langlebiger Hintergrunddienst in der.NET MAUI-Anwendung ist dafür verantwortlich, die lokalen Änderungen in die Cloud zu übertragen. Dieser Dienst prüft in regelmäßigen Abständen oder bei Wiederherstellung der Konnektivität die lokalen Tabellen auf Einträge mit is\_synced = false. Diese ausstehenden Änderungen werden gesammelt und über die supabase-csharp-API an das Backend gesendet. Nach erfolgreicher Bestätigung durch den Server wird das is\_synced-Flag in der lokalen Datenbank auf true gesetzt.

#### Konfliktlösung

Für den Anwendungsfall dieser App ist eine "Last Write Wins"-Strategie (der letzte Schreibvorgang gewinnt) in der Regel ausreichend und am einfachsten zu implementieren. Da jeder Datensatz und jede Änderung mit einem genauen Zeitstempel versehen ist (logged\_at, updated\_at), kann im Falle eines Konflikts der jüngste Datensatz als maßgeblich betrachtet werden. Komplexere Strategien wie Conflict-free Replicated Data Types (CRDTs) bieten zwar mathematisch elegantere Lösungen für kollaborative Bearbeitungsszenarien, führen aber auch eine erhebliche Komplexität ein, die für die erste Version dieses Projekts nicht gerechtfertigt ist.42 Die gewählte Architektur mit dem Outbox-Pattern und Zeitstempeln bietet einen robusten und pragmatischen Ansatz für die Synchronisation.

### 3.3. Benutzerauthentifizierung und Datensicherheit

Ein sicheres und einfaches Authentifizierungssystem ist entscheidend für das Vertrauen der Nutzer und den Schutz ihrer Daten.

#### Authentifizierungsfluss

Die Benutzerauthentifizierung wird vollständig über **Supabase Auth** abgewickelt. Dies bietet standardmäßig sichere und skalierbare Mechanismen für:

* **Registrierung und Anmeldung per E-Mail und Passwort:** Der klassische Anmeldeweg.
* **Social Logins:** Integration von Anbietern wie Google, um den Anmeldeprozess für Benutzer zu vereinfachen.

Die supabase-csharp-Clientbibliothek übernimmt die gesamte Komplexität der Authentifizierung, einschließlich der sicheren Speicherung und automatischen Erneuerung von JSON Web Tokens (JWTs), die für jede API-Anfrage zur Identifizierung des Benutzers verwendet werden.26

#### Datenisolation durch Row Level Security (RLS)

Das Herzstück des Sicherheitskonzepts ist die konsequente Anwendung von **Row Level Security (RLS)** in der PostgreSQL-Datenbank.18 Für jede Tabelle, die benutzerspezifische Daten enthält, werden RLS-Richtlinien (Policies) definiert. Diese Richtlinien sind SQL-Regeln, die auf Datenbankebene für jede einzelne Datenzeile und für jede Operation (SELECT, INSERT, UPDATE, DELETE) durchgesetzt werden.

**Beispiel einer RLS-Richtlinie für die workout\_sessions-Tabelle:**

SQL

CREATE POLICY "Benutzer können nur ihre eigenen Trainingseinheiten sehen."  
ON public.workout\_sessions  
FOR SELECT  
USING (auth.uid() = user\_id);  
  
CREATE POLICY "Benutzer können nur für sich selbst Trainingseinheiten erstellen."  
ON public.workout\_sessions  
FOR INSERT  
WITH CHECK (auth.uid() = user\_id);

Diese Richtlinien stellen sicher, dass ein authentifizierter API-Aufruf, der versucht, auf die workout\_sessions-Tabelle zuzugreifen, automatisch so gefiltert wird, dass nur die Zeilen zurückgegeben (oder eingefügt) werden, bei denen die user\_id-Spalte mit der ID des authentifizierten Benutzers (auth.uid()) übereinstimmt. Dies macht es auf Datenbankebene unmöglich, dass Daten eines Benutzers versehentlich oder böswillig an einen anderen Benutzer gelangen, und bietet eine extrem starke Sicherheitsgarantie.19

### 3.4. Engineering des Workout-Lebenszyklus

Dieser Abschnitt beschreibt die technische Umsetzung der zentralen Benutzerinteraktionen, von der Planung bis zur Durchführung eines Trainings.

#### Workout-Planer

Diese Funktion ermöglicht es Benutzern, wiederverwendbare Trainingsvorlagen zu erstellen und zu verwalten.

* **Funktionalität:** Benutzer können neue workout\_plans erstellen, ihnen einen Namen geben und eine geordnete Liste von exercises aus der Stammdatenbank hinzufügen. Bestehende Pläne können bearbeitet (Übungen hinzufügen/entfernen/neu anordnen) und gelöscht werden.
* **Implementierung:** Technisch gesehen handelt es sich hierbei um einfache CRUD-Operationen (Create, Read, Update, Delete) auf den lokalen SQLite-Tabellen workout\_plans und plan\_exercises. Die Änderungen werden über den in Abschnitt 3.2 beschriebenen Synchronisationsmechanismus an das Backend weitergegeben.

#### Live-Session-Tracker

Dies ist der interaktivste und wichtigste Bildschirm der Anwendung während des Trainings. Das Design muss auf eine schnelle, reibungslose und ablenkungsfreie Dateneingabe optimiert sein, inspiriert von den klaren Schnittstellen führender Apps wie Strong und Hevy.45

* **Start eines Workouts:** Der Benutzer wählt einen seiner workout\_plans aus oder startet ein leeres Training. Daraufhin wird lokal ein neuer Eintrag in der workout\_sessions-Tabelle mit einem started\_at-Zeitstempel erstellt.
* **Protokollierung eines Satzes:** Für jede Übung im Plan zeigt die UI Eingabefelder für Gewicht und Wiederholungen an. Wichtig ist, dass die App die Leistung des letzten Satzes derselben Übung aus einer früheren Einheit anzeigt, um das Prinzip der progressiven Überlastung zu unterstützen. Nach der Eingabe und Bestätigung durch den Benutzer wird ein neuer Eintrag in der lokalen set\_logs-Tabelle erstellt, der mit der aktuellen session\_id und exercise\_id verknüpft ist.
* **Abschluss eines Workouts:** Wenn der Benutzer das Training beendet, wird der completed\_at-Zeitstempel in der workout\_sessions-Tabelle gesetzt.

#### Der dynamische Pausentimer

Die Anforderung eines integrierten Timers mit festen oder zufälligen Intervallen zwischen 1 und 3 Minuten ist ein zentrales Feature.

* **Auslöser:** Der Timer startet automatisch, sobald der Benutzer einen Satz als abgeschlossen markiert.
* **Konfiguration:** In den App-Einstellungen kann der Benutzer zwischen den Modi "Fest" und "Zufällig" wählen und für den festen Modus eine Standarddauer (z.B. 90 Sekunden) festlegen.
* **Implementierungslogik:** Die Logik kann in einer einfachen Funktion im ViewModel des Live-Trackers gekapselt werden:  
  C#  
  private void OnSetCompleted()  
  {  
   int durationInSeconds;  
   var userSettings = \_settingsService.GetTimerSettings();  
    
   if (userSettings.TimerMode == TimerMode.Fixed)  
   {  
   durationInSeconds = userSettings.FixedDurationSeconds;  
   }  
   else // TimerMode.Random  
   {  
   Random random = new Random();  
   durationInSeconds = random.Next(60, 181); // Zufällige Zahl zwischen 60 und 180 Sekunden  
   }  
    
   \_timerService.StartCountdown(durationInSeconds);  
  }
* **Benachrichtigung:** Nach Ablauf des Timers erhält der Benutzer eine deutliche Benachrichtigung, idealerweise sowohl akustisch als auch durch Vibration (haptisches Feedback), um den Beginn des nächsten Satzes zu signalisieren.

## IV. User Experience und Interface (UI/UX) Designstrategie

Eine herausragende User Experience (UX) ist entscheidend für den Erfolg einer Fitness-App. Die Anwendung muss nicht nur funktional sein, sondern auch intuitiv, motivierend und visuell ansprechend. Das Design sollte den Benutzer dabei unterstützen, seine Ziele zu erreichen und langfristig engagiert zu bleiben.49

### 4.1. Die Onboarding-Reise

Der erste Eindruck ist entscheidend. Der Onboarding-Prozess muss den Benutzer schnell an die Hand nehmen, den Wert der App demonstrieren und ihn für die Nutzung personalisieren, ohne ihn mit zu vielen Schritten zu überfordern.50

* **Ziel:** Den Benutzer schnell aktivieren, indem der Nutzen der App klar kommuniziert und eine grundlegende Personalisierung vorgenommen wird. Der Prozess sollte kurz, prägnant und optional überspringbar sein.52
* **Vorgeschlagene Schritte:**
  1. **Willkommensbildschirme (1-3 Screens):** Prägnante Darstellung der Kernversprechen der App: "Plane deine Workouts", "Verfolge deinen Fortschritt", "Synchronisiere nahtlos". Ansprechende Grafiken und minimale Texte sind hier entscheidend.
  2. **Anmeldung/Registrierung:** Bieten Sie einfache Optionen wie E-Mail/Passwort und einen "Mit Google anmelden"-Button an, um die Hürde so niedrig wie möglich zu halten.50
  3. **Essentielle Personalisierung (2-3 Screens):** Fragen Sie nach den wichtigsten Informationen, um die Erfahrung von Anfang an anzupassen 53:
     + Was ist dein primäres Fitnessziel? (z.B. Kraftaufbau, Muskelaufbau, Gewichtsverlust)
     + Wie würdest du dein Erfahrungslevel beschreiben? (Anfänger, Fortgeschritten, Experte)
     + Welche Gewichtseinheit bevorzugst du? (kg / lbs)  
       Diese Daten werden verwendet, um möglicherweise erste Trainingsplan-Vorschläge zu machen oder die Benutzeroberfläche anzupassen.
  4. **Kurze, optionale Einführungstour:** Anstatt den Benutzer zu zwingen, kann ein optionaler "Zeig mir, wie es geht"-Button eine kurze, interaktive Tour durch die Hauptfunktionen starten: wie man einen Plan erstellt und ein Training beginnt.54

### 4.2. Das Dashboard: Visualisierung des Fortschritts

Das Dashboard ist die zentrale Anlaufstelle für den Benutzer, um seine Aktivitäten und Fortschritte auf einen Blick zu erfassen. Es muss motivierend wirken und klare, handlungsorientierte Einblicke bieten.55 Das Design muss responsiv sein und sich von einem mehrspaltigen Layout auf dem Windows-PC elegant in eine scrollbare Ansicht auf dem Android-Gerät verwandeln.

* **Ziel:** Eine schnelle und motivierende Übersicht über die wichtigsten Fitness-Metriken zu geben.
* **Schlüsselvisualisierungen:**
  + **Aktivitäts-Heatmap:** Eine Kalenderansicht, ähnlich der Beitragsgrafik von GitHub, in der jeder Tag, an dem ein Training absolviert wurde, farblich markiert wird. Dies visualisiert die Konsistenz des Benutzers auf eindrucksvolle Weise.55
  + **Key Performance Indicators (KPIs):** Große, gut sichtbare Zahlen für die wichtigsten Gesamtstatistiken, die sofort ins Auge fallen.55
  + **Fortschrittsdiagramme:** Interaktive Liniendiagramme, die die Kraftentwicklung bei Schlüsselübungen im Zeitverlauf darstellen. Dies könnte beispielsweise das geschätzte 1-Rep-Max (Maximalkraft) für Bankdrücken, Kniebeugen und Kreuzheben sein. Diese Daten werden durch die Analyse der set\_logs-Tabelle generiert.57

| Dashboard-Widget | Beschreibung | Zweck | Datenquelle(n) |
| --- | --- | --- | --- |
| **Aktivitäts-Heatmap** | Jahreskalender mit farblich markierten Trainingstagen. | Visualisierung der Trainingskonsistenz und -frequenz. | workout\_sessions (Datum) |
| **KPI: Gesamt-Workouts** | Große Zahl, die alle absolvierten Workouts anzeigt. | Schneller Überblick über die Gesamtaktivität. | workout\_sessions (COUNT) |
| **KPI: Gesamtvolumen** | Gesamtes bewegtes Gewicht (Sätze \* Wdh. \* Gewicht). | Messung des gesamten Arbeitsaufwands. | set\_logs (SUM) |
| **KPI: Workout-Streak** | Anzahl der aufeinanderfolgenden Wochen mit mind. einem Workout. | Förderung der Regelmäßigkeit und Gewohnheitsbildung. | workout\_sessions (Datum) |
| **Fortschrittsdiagramm** | Liniendiagramm für ausgewählte Übungen (z.B. 1RM-Schätzung). | Visualisierung der Kraftsteigerung über die Zeit. | set\_logs (Gewicht, Wdh., Datum) |

### 4.3. Motivationssteigerung durch Gamification

Gamification nutzt spieltypische Elemente in einem nicht-spielerischen Kontext, um die Motivation zu steigern und die Benutzerbindung zu erhöhen. In einer Fitness-App kann dies den entscheidenden Unterschied machen, ob ein Benutzer langfristig dabei bleibt oder die App nach kurzer Zeit wieder aufgibt.58

* **Ziel:** Durch subtile, aber wirkungsvolle psychologische Anreize die Bildung gesunder Gewohnheiten fördern und die Nutzung der App belohnen.61
* **Vorgeschlagene Funktionen:**
  + **Persönliche Rekorde (PRs):** Wenn ein Benutzer einen Satz protokolliert, der einen neuen persönlichen Rekord darstellt (z.B. das höchste Gewicht für eine bestimmte Wiederholungszahl bei einer Übung), sollte die Benutzeroberfläche dies sofort mit einer kleinen Animation, einem Konfetti-Effekt oder einem "PR!"-Badge feiern. Dies liefert sofortiges positives Feedback und verstärkt das Erfolgserlebnis.46
  + **Workout-Streaks:** Das Dashboard und der Profilbildschirm sollten die aktuelle "Streak" (Serie) von aufeinanderfolgenden Wochen mit mindestens einem Training prominent anzeigen. Das menschliche Bestreben, eine solche Serie nicht zu unterbrechen, ist ein starker Motivator.62
  + **Erfolgs-Badges:** Verleihen Sie virtuelle Abzeichen für das Erreichen bestimmter Meilensteine. Diese können in einer Trophäensammlung im Profil des Benutzers angezeigt werden. Beispiele:
    - "Trainingsstarter" (Erstes Workout abgeschlossen)
    - "Konstanz-König" (10 Workouts abgeschlossen)
    - "Schwergewicht" (Insgesamt 10.000 kg Volumen gehoben)
    - "Marathon-Mann" (50 Workouts abgeschlossen)  
      Diese Badges dienen als sichtbare Anerkennung für die erbrachte Leistung und fördern das langfristige Engagement.58

## V. Entwicklungs-, Optimierungs- und Bereitstellungs-Roadmap

Dieser letzte Abschnitt skizziert den technischen Fahrplan für die Konstruktion, das Testen und die Veröffentlichung der Anwendung. Er legt bewährte Entwicklungsmuster fest, definiert eine proaktive Strategie zur Leistungsoptimierung und stellt eine Checkliste für die Einreichung in den App Stores bereit.

### 5.1. Entwicklungsmuster und -praktiken (.NET MAUI)

Um eine saubere, testbare und wartbare Codebasis zu gewährleisten, ist die Anwendung etablierter Architekturmuster unerlässlich.

* **MVVM-Architektur (Model-View-ViewModel):** Das MVVM-Muster ist der De-facto-Standard für die Entwicklung von XAML-basierten Anwendungen. Es trennt die Benutzeroberfläche (View) strikt von der Präsentationslogik und den Daten (ViewModel) und der Geschäftslogik (Model).64 Diese Trennung ermöglicht es, die Logik unabhängig von der UI zu entwickeln und zu testen, was die Wartbarkeit und Wiederverwendbarkeit des Codes erheblich verbessert.
* **CommunityToolkit.Mvvm:** Um die Implementierung des MVVM-Musters zu vereinfachen und die Menge an sich wiederholendem Code (Boilerplate) drastisch zu reduzieren, wird die von Microsoft unterstützte Bibliothek CommunityToolkit.Mvvm eingesetzt.65 Dieses Toolkit nutzt C#-Source-Generatoren, um automatisch die notwendige Infrastruktur für datenbindungsfähige Eigenschaften und Befehle zu erstellen.
  + [ObservableProperty]: Dieses Attribut wird über einem privaten Feld platziert und generiert zur Kompilierzeit eine öffentliche Eigenschaft, die automatisch INotifyPropertyChanged-Benachrichtigungen auslöst, wenn sich ihr Wert ändert.68
  + ``: Dieses Attribut wird über einer Methode platziert und generiert eine ICommand-Implementierung, die direkt von der XAML-UI gebunden werden kann (z.B. an einen Button-Klick).68
* **Dependency Injection (DI):**.NET MAUI verfügt über einen integrierten Host-Builder, der Dependency Injection unterstützt. Alle Dienste, wie z.B. die Datenbankzugriffsklasse, die Synchronisationsdienste und die ViewModels, werden beim Start der Anwendung in einem DI-Container registriert. Dies ermöglicht eine lose Kopplung der Komponenten, was die Testbarkeit und Flexibilität der Architektur weiter erhöht.35

### 5.2. Plan zur Leistungsoptimierung

Die Performance einer mobilen Anwendung ist kein nachträgliches Feature, sondern eine grundlegende Anforderung. Eine langsame Startzeit oder eine ruckelnde Benutzeroberfläche sind Hauptgründe für die Deinstallation einer App.71 Daher muss die Optimierung von Anfang an ein zentraler Bestandteil des Entwicklungsprozesses sein. Die Analyse der.NET MAUI-Plattform zeigt, dass die größten Leistungsengpässe typischerweise beim Start der App und beim Rendern von Listen auftreten.71

* **Schnelle Startzeit:**
  + **Ahead-of-Time (AOT) Kompilierung:** Für alle Release-Builds, insbesondere für Android und iOS, wird die AOT-Kompilierung aktiviert. AOT übersetzt den.NET-Code bereits während des Build-Prozesses in nativen Maschinencode, was die Notwendigkeit der Just-in-Time (JIT) Kompilierung zur Laufzeit eliminiert und die Startzeit der App signifikant verkürzt.71
  + **Linker:** Der.NET-Linker wird im TrimMode=link-Modus für Release-Builds konfiguriert. Er analysiert die Anwendung und entfernt ungenutzten Code aus den Assemblies, was die Gesamtgröße der App reduziert und somit zu schnelleren Downloads und einer kürzeren Ladezeit beiträgt.71
  + **XAML-Kompilierung (XAMLc):** XAMLc ist standardmäßig aktiviert und wandelt XAML-Dateien zur Kompilierzeit in Code um. Dies vermeidet das Parsen von XML zur Laufzeit und beschleunigt das Laden von Seiten erheblich.75
* **Flüssige Benutzeroberfläche:**
  + **UI-Virtualisierung mit CollectionView:** Für alle Listen in der Anwendung (z.B. Übungslisten, Trainingsverlauf) wird die CollectionView verwendet. Sie ist für die Darstellung großer Datenmengen optimiert, da sie nur die Elemente rendert, die aktuell auf dem Bildschirm sichtbar sind (UI-Virtualisierung).77
  + **Optimierung von CollectionView-Item-Templates:** Die häufigste Ursache für schlechte CollectionView-Performance sind komplex verschachtelte Layouts innerhalb der DataTemplate. Statt eines StackLayout, das aufwendige Messungen erfordert, wird ein Grid mit fest definierten Zeilen- und Spaltenhöhen verwendet. Dies ermöglicht es dem Layout-System, die Größe jedes Elements sofort zu berechnen, was das Scrollen dramatisch beschleunigt.73
  + **Kompilierte Bindungen:** In allen XAML-Views wird das x:DataType-Attribut gesetzt, um kompilierte Datenbindungen zu aktivieren. Im Gegensatz zu klassischen Bindungen, die zur Laufzeit auf Reflection angewiesen sind, werden kompilierte Bindungen zur Kompilierzeit aufgelöst und sind um ein Vielfaches schneller.75
  + **Bildoptimierung:** Alle in der App verwendeten Bilder werden im modernen .webp-Format bereitgestellt, das eine hohe Kompression bei guter Qualität bietet. Zudem wird sichergestellt, dass die Bilder in der richtigen Auflösung für ihre Anzeigecontainer vorliegen, um unnötigen Speicherverbrauch und Skalierungsaufwand zur Laufzeit zu vermeiden.71

### 5.3. Ein Rahmenwerk für das Testen

Ein umfassendes Testkonzept ist entscheidend, um die Qualität und Stabilität der Anwendung sicherzustellen.

* **Unit-Tests:** Die Verwendung des MVVM-Musters isoliert die gesamte Geschäfts- und Präsentationslogik in den ViewModels. Dies macht sie ideal für Unit-Tests. Mit einem Test-Framework wie xUnit oder NUnit werden Tests geschrieben, die die Logik der Commands und die Zustandsänderungen der Eigenschaften in den ViewModels überprüfen. Dies kann vollständig ohne die Ausführung der UI geschehen und stellt sicher, dass die Kernlogik der Anwendung korrekt funktioniert.65
* **Integrationstests:** Das größte Risiko in der Anwendungsarchitektur ist die Offline-Synchronisationslogik. Es werden spezielle Integrationstests entwickelt, die dieses System ganzheitlich prüfen. Diese Tests simulieren verschiedene Szenarien:
  1. Erstellen von Daten im Online-Modus.
  2. Wechsel in den Offline-Modus.
  3. Ändern und Erstellen weiterer Daten offline.
  4. Wiederherstellen der Online-Verbindung.
  5. Überprüfung, ob alle lokalen Änderungen korrekt an den Server übertragen und serverseitige Änderungen korrekt auf den Client heruntergeladen werden.

### 5.4. Go-to-Market: Checkliste für die App-Store-Einreichung

Die finale Phase des Projekts ist die Veröffentlichung in den jeweiligen App Stores. Eine sorgfältige Vorbereitung ist hierbei unerlässlich.

#### Google Play Store (Android)

* **Entwicklerkonto:** Erstellen eines Google Play Developer-Kontos (einmalige Gebühr von 25 USD).80
* **Store-Listing-Assets:**
  + App-Icon: Hochauflösend, 512x512 Pixel.81
  + Feature-Grafik: 1024x500 Pixel, dient als Werbebanner im Store.82
  + Screenshots: Mindestens 2, maximal 8 hochwertige Screenshots, die die Kernfunktionen der App auf einem Android-Gerät zeigen.81
* **Metadaten:**
  + App-Name: Maximal 50 Zeichen.81
  + Kurzbeschreibung: Maximal 80 Zeichen, entscheidend für den ersten Eindruck.80
  + Vollständige Beschreibung: Detaillierte Beschreibung der Funktionen, maximal 4000 Zeichen.81
* **Compliance:**
  + Datenschutzrichtlinie: Bereitstellung einer URL zu einer gültigen Datenschutzrichtlinie ist obligatorisch, insbesondere da Gesundheitsdaten verarbeitet werden.81
  + Inhaltsbewertung: Ausfüllen des Fragebogens zur Altersfreigabe.82
  + Einhaltung aller Google Play-Richtlinien, insbesondere in Bezug auf Benutzerdaten, Gesundheit und Fitness.83

#### Microsoft Store (Windows)

* **Entwicklerkonto:** Erstellen eines Entwicklerkontos im Microsoft Partner Center.84
* **Anwendungspaket:** Erstellen eines signierten MSIX-Pakets aus dem.NET MAUI-Projekt. MSIX ist das moderne Paketformat für Windows-Anwendungen.84
* **Store-Listing-Assets:**
  + App-Name: Muss im Partner Center reserviert werden und sollte originell und kurz sein.84
  + Beschreibung: Klare und ansprechende Beschreibung der App-Funktionen.86
  + Screenshots: Mindestens ein Screenshot ist erforderlich, um die Windows-Anwendung in Aktion zu zeigen.86
  + Store-Logo und andere Grafiken gemäß den Spezifikationen.87
* **Metadaten und Konfiguration:**
  + Kategorie: Auswahl der passenden Kategorie (z.B. "Gesundheit & Fitness").
  + Datenschutzrichtlinie: Angabe der URL zur Datenschutzrichtlinie.87
  + Preise und Verfügbarkeit: Festlegung der Märkte und des Preises (in diesem Fall kostenlos).87

Durch die Befolgung dieser detaillierten Roadmap kann die Entwicklung der Fitnessanwendung strukturiert, effizient und mit einem klaren Fokus auf Qualität, Performance und Benutzererfahrung erfolgen.

#### Referenzen

1. Which cross-platform app development framework to invest in? - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/learnprogramming/comments/1itpdjf/which_crossplatform_app_development_framework_to/>
2. Cross-Platform Mobile Development 2025: Which Framework Should You Bet On?, Zugriff am August 25, 2025, <https://www.orientsoftware.com/blog/cross-platform-mobile-development/>
3. .NET Multi-platform App UI (.NET MAUI) | .NET, Zugriff am August 25, 2025, <https://dotnet.microsoft.com/en-us/apps/maui>
4. Free .NET MAUI eBook to Support You in Building Cross-Platform Applications - Transparity, Zugriff am August 25, 2025, <https://www.transparity.com/app-innovation/free-net-maui-ebook/>
5. Get started with Windows development using WinUI - Windows apps | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/get-started/start-here>
6. System requirements for Windows app development, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/windows-app-sdk/system-requirements>
7. Top 10 Cross-Platform App Development Frameworks (2025) - Apptunix, Zugriff am August 25, 2025, <https://www.apptunix.com/blog/cross-platform-app-development-frameworks/>
8. Top 8 Cross-Platform App Development Frameworks in 2025 - PixelPlex, Zugriff am August 25, 2025, <https://pixelplex.io/blog/cross-app-development-frameworks/>
9. Best Backend-as-a-Service Software in 2025 | 6sense, Zugriff am August 25, 2025, <https://6sense.com/tech/backend-as-a-service>
10. 21 Best MBaaS Software Picks of 2025 - The CTO Club, Zugriff am August 25, 2025, <https://thectoclub.com/tools/best-mbaas-software/>
11. Best Backend as a Service (BaaS) Platforms - Galaxy, Zugriff am August 25, 2025, <https://www.getgalaxy.io/learn/data-tools/best-backend-as-a-service-baas-platforms>
12. BaaS Providers - 20 Backend Platforms in 2025 - Back4App Blog, Zugriff am August 25, 2025, <https://blog.back4app.com/baas-providers/>
13. Supabase vs Firebase, Zugriff am August 25, 2025, <https://supabase.com/alternatives/supabase-vs-firebase>
14. Supabase | The Postgres Development Platform., Zugriff am August 25, 2025, <https://supabase.com/>
15. Supabase vs Firebase: Best BaaS Platform for 2025 - The Brihaspati Infotech, Zugriff am August 25, 2025, <https://www.brihaspatitech.com/blog/supabase-vs-firebase-which-backend-should-you-choose-in-2025/>
16. Firebase vs Supabase - DEV Community, Zugriff am August 25, 2025, <https://dev.to/codeparrot/firebase-vs-supabase-4770>
17. Supabase vs. Firebase: Which is best? [2025] - Zapier, Zugriff am August 25, 2025, <https://zapier.com/blog/supabase-vs-firebase/>
18. Securing your API | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/guides/api/securing-your-api>
19. Best Security Practices in Supabase: A Comprehensive Guide - Supadex, Zugriff am August 25, 2025, <https://www.supadex.app/blog/best-security-practices-in-supabase-a-comprehensive-guide>
20. Supabase vs Firebase: Choosing the Right Backend for Your Next Project - Jake Prins, Zugriff am August 25, 2025, <https://www.jakeprins.com/blog/supabase-vs-firebase-2024>
21. Bringing offline-first to Supabase, the right way - PowerSync, Zugriff am August 25, 2025, <https://www.powersync.com/blog/bringing-offline-first-to-supabase>
22. Building offline-first mobile apps with Supabase, Flutter and Brick - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/Supabase/comments/1kpidhi/building_offlinefirst_mobile_apps_with_supabase/>
23. Building offline-first mobile apps with Supabase, Flutter and Brick, Zugriff am August 25, 2025, <https://supabase.com/blog/offline-first-flutter-apps>
24. Using Supabase offline #357 - GitHub, Zugriff am August 25, 2025, <https://github.com/orgs/supabase/discussions/357>
25. C#: Introduction | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/reference/csharp/introduction>
26. C#: Initializing | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/reference/csharp/initializing>
27. C#: Column is equal to a value | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/reference/csharp/eq>
28. Supabase Crash Course For .NET Developers: Building an API From Scratch - YouTube, Zugriff am August 25, 2025, <https://www.youtube.com/watch?v=uviVTDtYeeE>
29. csharp.txt - Supabase, Zugriff am August 25, 2025, <https://supabase.com/llms/csharp.txt>
30. C#: Subscribe to channel | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/reference/csharp/subscribe>
31. Designing a fitness / weight lifiting routine database - Software Engineering Stack Exchange, Zugriff am August 25, 2025, <https://softwareengineering.stackexchange.com/questions/226189/designing-a-fitness-weight-lifiting-routine-database>
32. How to Design a Database for Health and Fitness Tracking Applications - GeeksforGeeks, Zugriff am August 25, 2025, <https://www.geeksforgeeks.org/dbms/how-to-design-a-database-for-health-and-fitness-tracking-applications/>
33. Database Schema for a Gym Exercise Log App - Stack Overflow, Zugriff am August 25, 2025, <https://stackoverflow.com/questions/54220956/database-schema-for-a-gym-exercise-log-app>
34. Easily Bind SQLite Data to .NET MAUI ListView and Perform CRUD Actions - Syncfusion, Zugriff am August 25, 2025, <https://www.syncfusion.com/blogs/post/sqlite-data-to-net-maui-listview>
35. NET MAUI local databases - Microsoft Community, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/dotnet/maui/data-cloud/database-sqlite?view=net-maui-9.0>
36. Exploring SQLite Integration in .NET MAUI | by Erdal Kama - Medium, Zugriff am August 25, 2025, <https://medium.com/@erdalkama/exploring-sqlite-integration-in-net-maui-59371a8ec1d3>
37. Health apps: data protection and data security - Oppenhoff, Zugriff am August 25, 2025, <https://www.oppenhoff.eu/en/news/detail/health-apps-data-protection-and-data-security/>
38. SQLCipher for .NET - Full Database Encryption for SQLite on Windows, iOS, Android, macOS, and Linux | Zetetic, Zugriff am August 25, 2025, <https://www.zetetic.net/sqlcipher/sqlcipher-for-dotnet/>
39. jfversluis/MauiEncryptedSqliteSample: Sample code to demonstrate how to work with a encrypted SQLite database in .NET MAUI (and Xamarin.Forms) - GitHub, Zugriff am August 25, 2025, <https://github.com/jfversluis/MauiEncryptedSqliteSample>
40. Encrypt Database with SQLite-net and .NET MAUI (or Xamarin.Forms!) - YouTube, Zugriff am August 25, 2025, <https://www.youtube.com/watch?v=O1UQfoh4710>
41. Realtime | Supabase Docs, Zugriff am August 25, 2025, <https://supabase.com/docs/guides/realtime>
42. sillsdev/harmony: C# CRDT Library for building offline first apps - GitHub, Zugriff am August 25, 2025, <https://github.com/sillsdev/harmony>
43. Going local-first with Automerge and Convex, Zugriff am August 25, 2025, <https://stack.convex.dev/automerge-and-convex>
44. You might not need a CRDT (Conflict-free Replicated Data Type) : r/rust - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/rust/comments/zk54qm/you_might_not_need_a_crdt_conflictfree_replicated/>
45. Exercise Log - Hevy - #1 Workout Tracker & Planner Gym Log App, Zugriff am August 25, 2025, <https://www.hevyapp.com/use-cases/exercise-log/>
46. Fitness Log - Hevy - #1 Workout Tracker & Planner Gym Log App, Zugriff am August 25, 2025, <https://www.hevyapp.com/use-cases/fitness-log/>
47. Strong - Workout Tracker & Gym Log, Zugriff am August 25, 2025, <https://www.strong.app/>
48. Strong Workout Tracker Gym Log | ScreensDesign, Zugriff am August 25, 2025, <https://screensdesign.com/showcase/strong-workout-tracker-gym-log>
49. Fitness App UI Design: Key Principles for Engaging Workout Apps - Stormotion, Zugriff am August 25, 2025, <https://stormotion.io/blog/fitness-app-ux/>
50. App Onboarding Guide - Top 10 Onboarding Flow Examples 2025 - UXCam, Zugriff am August 25, 2025, <https://uxcam.com/blog/10-apps-with-great-user-onboarding/>
51. Creating a Superb Fitness App Design: Best Practices - Perpetio, Zugriff am August 25, 2025, <https://perpet.io/blog/what-should-be-ui-ux-design-of-fitness-app/>
52. Hi Guys! I am working on my very first UI/UX project (Fitness app) and have made quite a few screens and i would really appreciate your feedback. : r/FigmaDesign - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/FigmaDesign/comments/1jna70u/hi_guys_i_am_working_on_my_very_first_uiux/>
53. Onboarding Experience Design for Health and Fitness App — FitHealth - Medium, Zugriff am August 25, 2025, <https://medium.com/@ankitsingh.iitb/onboarding-experience-for-health-and-fitness-app-fithealth-ca8538541979>
54. Best Mobile App Onboarding Examples in 2025 - Plotline, Zugriff am August 25, 2025, <https://www.plotline.so/blog/mobile-app-onboarding-examples>
55. 5 Best Fitness Dashboard Examples - Quantize Analytics Ltd, Zugriff am August 25, 2025, <https://www.quantizeanalytics.co.uk/fitness-dashboard-example/>
56. Dashboard Design: best practices and examples - Justinmind, Zugriff am August 25, 2025, <https://www.justinmind.com/ui-design/dashboard-design-best-practices-ux>
57. Exercise Data Visualization - Number Analytics, Zugriff am August 25, 2025, <https://www.numberanalytics.com/blog/exercise-data-visualization>
58. The ultimate gamification guide - Adjust, Zugriff am August 25, 2025, <https://www.adjust.com/resources/guides/app-gamification/>
59. Fitness Revolution: Gamification Techniques For Workout Adherence - The Octalysis Group, Zugriff am August 25, 2025, <https://octalysisgroup.com/2023/11/fitness-revolution-gamification-techniques-for-workout-adherence/>
60. The Influence of Gamification and Information Technology Identity on Postadoption Behaviors of Health and Fitness App Users: Empirical Study in the United States, Zugriff am August 25, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC8406121/>
61. How Gamification Makes Fitness Apps Fun and Engaging - Mindster, Zugriff am August 25, 2025, <https://mindster.com/mindster-blogs/gamification-fitness-apps-engagement/>
62. Gamification For Fitness Apps - Nudge, Zugriff am August 25, 2025, <https://www.nudgenow.com/blogs/gamify-your-fitness-apps>
63. Hevy - Workout Tracker & Planner Gym App Weight Lifting Routine Log, Zugriff am August 25, 2025, <https://www.hevyapp.com/>
64. Introduction to .NET MAUI: A Complete Beginner's Guide, Zugriff am August 25, 2025, <https://eluminoustechnologies.com/blog/net-maui-guide/>
65. Upgrade your app with MVVM concepts - .NET MAUI | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/dotnet/maui/tutorials/notes-mvvm?view=net-maui-9.0>
66. MVVM Fundamentals — Examples with .NET MAUI | by José Rojas - Medium, Zugriff am August 25, 2025, <https://medium.com/@rojasjimenezjosea/mvvm-fundamentals-examples-with-net-maui-5b0af71e99d6>
67. Introduction to the MVVM Toolkit - Community Toolkits for .NET | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/>
68. .NET MAUI with MVVM Toolkit - C# Corner, Zugriff am August 25, 2025, <https://www.c-sharpcorner.com/article/net-maui-with-mvvm-toolkit/>
69. MVVM with the MVVM Community Toolkit | My Memory, Zugriff am August 25, 2025, <https://putridparrot.com/blog/mvvm-with-the-mvvm-community-toolkit/>
70. A light tutorial of CommunityToolkit.Mvvm and Dependency Injection - FormatException, Zugriff am August 25, 2025, <https://formatexception.com/2024/03/a-light-tutorial-of-communitytoolkit-mvvm-and-dependency-injection/>
71. 10 Quick Performance-Optimizing Tricks for MAUI Apps [2025] - Net MAUI, Zugriff am August 25, 2025, <https://hire-maui-developer.com/blog/dotnet-maui-performance-tricks>
72. Performance Optimization in .NET MAUI: Boost Your App's Speed, Zugriff am August 25, 2025, <https://vlinkinfo.com/blog/performance-optimization-in-dot-net-maui>
73. CollectionView Struggles with MAUI Core : r/dotnetMAUI - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/dotnetMAUI/comments/1luk3yk/collectionview_struggles_with_maui_core/>
74. I want to improve the performance of my app. : r/dotnetMAUI - Reddit, Zugriff am August 25, 2025, <https://www.reddit.com/r/dotnetMAUI/comments/1c6u3xx/i_want_to_improve_the_performance_of_my_app/>
75. NET MAUI Performance Features in .NET 9 - Microsoft Developer Blogs, Zugriff am August 25, 2025, <https://devblogs.microsoft.com/dotnet/dotnet-9-performance-improvements-in-dotnet-maui/>
76. Improve app performance - .NET MAUI | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/dotnet/maui/deployment/performance?view=net-maui-9.0>
77. Boost .NET MAUI App Performance: Best Practices for Speed and Scalability - Syncfusion, Zugriff am August 25, 2025, <https://www.syncfusion.com/blogs/post/net-maui-high-performance-apps/amp>
78. 7 Tips to Boost .NET MAUI Mobile App Performance - Blogs - DevExpress, Zugriff am August 25, 2025, <https://community.devexpress.com/blogs/mobile/archive/2023/07/21/7-tips-to-boost-net-maui-mobile-app-performance.aspx>
79. Writing Unit Tests for your .NET MAUI app - Mark Allibone, Zugriff am August 25, 2025, <https://mallibone.com/post/maui-unit-testing>
80. How To Submit an Android App to the Google Play Store (2025 Guide) | InspiringApps, Zugriff am August 25, 2025, <https://www.inspiringapps.com/blog/how-to-submit-app-to-google-play-store>
81. Checklist For How To Submit App To Google Play Store - Twinr.dev, Zugriff am August 25, 2025, <https://twinr.dev/blogs/submit-app-to-google-play-store/>
82. Google Play Store - Publishing Checklist - BuddyBoss, Zugriff am August 25, 2025, <https://www.buddyboss.com/resources/wp-content/uploads/2025/01/Google-Playstore-Publishing-Checklist.pdf>
83. Developer Policy Center - Google Play, Zugriff am August 25, 2025, <https://play.google/developer-content-policy/>
84. Publish Windows apps and games to Microsoft Store, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/publish/>
85. packaging, deployment, and process - Windows apps | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/get-started/intro-pack-dep-proc>
86. Submit your app to Microsoft Store - Windows apps, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/publish/faq/submit-your-app>
87. Create an app submission for your MSIX app - Windows apps | Microsoft Learn, Zugriff am August 25, 2025, <https://learn.microsoft.com/en-us/windows/apps/publish/publish-your-app/msix/create-app-submission>