**Frontend-Technologie: Flutter**

**Kontext und Problemstellung**  
Die Fitness-App soll sowohl auf Android- als auch auf iOS-Geräten verfügbar sein. Ziel ist es, eine einheitliche Benutzeroberfläche zu schaffen, die auf beiden Plattformen konsistent funktioniert, ohne doppelten Entwicklungsaufwand für zwei separate native Apps.  
**Frage:**  
Welche Technologie eignet sich am besten, um mit einem einzigen Codebase eine performante und plattformübergreifende App für Android und iOS zu entwickeln?

**Betrachtete Varianten**

* **Variante 1:** Flutter (Cross-Plattform mit einer Codebasis für Android/iOS)
* **Variante 2:** Native Entwicklung für Android (Kotlin) und iOS (Swift)
* **Variante 3:** React Native (Cross-Plattform mit nativer Bridge)

**Entscheidung**  
**Gewählte Variante:** *Flutter*  
Flutter ermöglicht es, mit einer einzigen Codebasis eine moderne, reaktive Benutzeroberfläche für Android und iOS zu entwickeln. Es bietet eine sehr gute Performance und flexible UI-Komponenten für ansprechendes Design.

**Begründung:**

* **Variante 1 (Flutter):**  
  **Vorteile:**
  + Eine Codebasis für beide Plattformen → weniger Entwicklungs- und Wartungsaufwand
  + Sehr gute Performance durch native Kompilierung
  + Umfangreiche UI-Komponenten für ein konsistentes Look & Feel
  + Große Community und gute Dokumentation  
    **Nachteile:**
  + App-Größe etwas größer als bei nativen Apps
  + Zugriff auf einige native Funktionen erfordert zusätzliche Integration (über Plattform-Channels)
* **Variante 2 (Native Entwicklung):**  
  **Vorteile:**
  + Maximale Kontrolle über Plattform-spezifische Funktionen und UI
  + Optimale Performance  
    **Nachteile:**
  + Doppelte Entwicklung → höherer Zeit- und Kostenaufwand
  + Höhere Wartungskosten bei Änderungen
* **Variante 3 (React Native):**  
  **Vorteile:**
  + Ebenfalls eine gemeinsame Codebasis
  + Große Entwickler-Community  
    **Nachteile:**
  + Häufig auf native Brücken angewiesen → Performance kann darunter leiden
  + Weniger konsistente UI zwischen Plattformen ohne großen Anpassungsaufwand

**Status**  
**Angenommen**

**Konsequenzen**

* **Gut, weil:**
  + Schnellere Entwicklung durch gemeinsame Codebasis
  + Gleichmäßiges Nutzererlebnis auf Android und iOS
  + Gute UI/UX durch anpassbare Widgets und Hot Reload beim Entwickeln
  + Weniger Ressourcenbedarf für separate Plattformteams
* **Schlecht, weil:**
  + Eingeschränkter Zugriff auf neue native Plattformfeatures ohne zusätzliche Arbeit
  + Flutter-Apps sind initial etwas größer in der Dateigröße
  + Manchmal zusätzlicher Aufwand bei der Integration plattformspezifischer Bibliotheken