**Implementierung des Fragenpools**

* Erfolgt in Supabase
  + Zugang fehlt
* Für einen ersten Test ist unter src\data\mockQuesitions.js eine Datei mit Fragen hinterlegt
* Anbindung an die Anwendung erfolgt über die api questions.js oder questions.ts

**JavaScript (JS) für die API**

**Vorteile:**

* Direkt nutzbar in bestehenden .jsx-Komponenten, keine Umstellung nötig.
* Einfach zu schreiben und zu verstehen, kein zusätzlicher Lernaufwand.
* Kein Build-Step für TypeScript nötig, läuft sofort.
* Flexibel, besonders bei dynamischen Datenstrukturen.

**Nachteile:**

* Keine Typprüfung → Fehler wie falsche Feldnamen oder fehlende Daten fallen erst zur Laufzeit auf.
* Weniger Autovervollständigung und Code-Sicherheit in der IDE.
* Supabase-Datenstrukturen müssen manuell dokumentiert oder überprüft werden.

**TypeScript (TS) für die API**

**Vorteile:**

* **Typprüfung:** Stellt sicher, dass alle Daten die erwartete Struktur haben (z. B. Question).
* **Autovervollständigung:** IDE schlägt Typen, Felder und Methoden automatisch vor.
* **Fehlerfrüherkennung:** Viele Bugs werden schon beim Kompilieren gefunden.
* **Supabase-ready:** Einfache Integration mit Typsicherheit für Tabellen und Daten.
* **Dokumentation:** Interface wie Question dient als klare Spezifikation für das Team.

**Nachteile:**

* Braucht TypeScript-Kenntnisse und ggf. Umstellung von JS → TS.
* Zusätzlicher Build-Step (TS → JS), wenn noch nicht eingerichtet.
* Bei falschen Imports oder Typdefinitionen können anfänglich Kompilierungsfehler auftreten.

**Kurz gesagt:**

| **Option** | **Empfehlung für jetzt** | **Empfehlung für langfristig** |
| --- | --- | --- |
| JS | Schnell, funktioniert sofort | Gut, wenn das Team keine TS-Einführung plant |
| TS | Etwas Mehraufwand, sauberere API | Sehr gut, besonders bei Supabase & größeren Projekten |

**Implementierung der Spielfunktion**