ShazaPiano — Le Chemin Ultra Simple (FAIS ÇA)

Tout est décidé pour toi. Suis les étapes, copie-colle dans Cursor. Android seul. Backend local. IAP Android 1 $.

1) Choix figés (pas de réflexion, c'est décidé)

- Plateforme: Android uniquement (pour l'instant).
- Backend: FastAPI en local sur ton PC (aucun déploiement cloud).
- Audio piano: DÉSACTIVÉ (vidéos muettes) pour simplicité et stabilité.
- Previews 16 s: TRONQUÉES côté serveur (moins contournable).
- Practice Mode: monophonique (détection d'une note à la fois) au micro.
- Monétisation: 1 seul achat non■consommable Android (1 $) qui débloque tout.
- Firebase: Auth anonyme + Firestore (stocke l'achat).
- UI: thème sombre, style Shazam (sans copier).

2) Ordre exact des actions (fais dans cet ordre)

1 Créer le dossier du projet (monorepo).

2 Lancer le backend en local (FastAPI) en 1 commande.

3 Lancer l'app Flutter en mode dev (Android).

4 Générer les 4 previews (16 s) à partir d'un enregistrement.

5 Ajouter l'IAP Android (1 $) pour débloquer les versions complètes.

6 Activer le Practice Mode (détection fausses notes).

3) Commandes (copie-colle TELLES QUELLES)

```
# 3.1 — Créer les dossiers mkdir -p shazapiano/app shazapiano/backend shazapiano/docs && cd shazapiano && git init # 3.2 — Backend local (Python 3.11 recommandé) cd backend python -m venv .venv && source .venv/bin/activate pip install fastapi uvicorn[standard] numpy scipy soundfile librosa pretty_midi moviepy python-multipart torch basic-pitch # Crée un fichier app.py minimal (voir bloc plus bas), puis lance: uvicorn app:app --reload --host 0.0.0.0 --port 8000 # 3.3 — App Flutter cd ../app flutter create --org com.ludo --project-name shazapiano . flutter pub add dio permission_handler record video_player go_router flutter_riverpod shared_preferences firebase_core firebase_auth cloud_firestore in_app_purchase flutter run --dart-define=BACKEND_BASE=http://10.0.2.2:8000
```

4) Backend prêt à coller (ultra minimal)

```
# backend/app.py — COLLE TEL QUEL (version muette, previews 16 s, 4 niveaux) from fastapi import FastAPI, UploadFile, File from fastapi.responses import JSONResponse from fastapi.staticfiles import StaticFiles from pathlib import Path import uuid, subprocess, json app = FastAPI() BASE = Path(__file__).parent IN_DIR = BASE / "media" / "in"; OUT_DIR = BASE / "media" / "out" IN_DIR.mkdir(parents=True,
```

```
exist_ok=True); OUT_DIR.mkdir(parents=True, exist_ok=True)
@app.get("/health") def health(): return {"ok": True} def run(cmd):
subprocess.run(cmd, check=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE) def to_wav16k(src, dst):
run(["ffmpeg","-y","-i",str(src),"-ar","16000","-ac","1",str(dst)]) def
basicpitch_to_midi(wav, mid): # Appelle basic-pitch en ligne de commande
(simple) run(["basic-pitch","predict",str(wav), "--midi", str(mid),
"--no_audio"]) def render_keyboard(mid, mp4_full, mp4_prev): # Fausse
vidéo (placeholder) ultra simple pour POC (fond noir + titre) # Remplace
plus tard par rendu clavier — ici on tronque à 16s pour preview. # Full
(30s noir) et preview (16s noir) pour que l'app puisse tester le flux. run
(["ffmpeg","-y","-f","lavfi","-i","color=c=black:s=1280x360:d=30","-c:v","
libx264",str(mp4_full)]) run(["ffmpeg","-y","-t","16","-i",str(mp4_full),"
-c:v","libx264",str(mp4_prev)]) @app.post("/process") async def
process(file: UploadFile = File(...)): uid = uuid.uuid4().hex src = IN_DIR
/ f"{uid}_{file.filename}" with open(src, "wb") as f: f.write(await
file.read()) wav = IN_DIR / f"{uid}.wav"; to_wav16k(src, wav) # 4 niveaux :
on génère 4 couples (full/preview) — ici même rendu (placeholder) out = []
for level in [1,2,3,4]: mid = OUT_DIR / f"{uid}_L{level}.mid" try:
basicpitch_to_midi(wav, mid) except Exception: # POC : s'il y a un souci
d'inférence, on crée quand même des fichiers vides valides
mid.write_bytes(b'') mp4_full = OUT_DIR / f"{uid}_L{level}_full.mp4"
mp4_prev = OUT_DIR / f"{uid}_L{level}_preview.mp4" render_keyboard(mid,
mp4_full, mp4_prev) out.append({ "level": level, "preview_url":
f"/media/out/{mp4_prev.name}", "video_url": f"/media/out/{mp4_full.name}",
"midi_url": f"/media/out/{mid.name}", "key_guess": "C", "tempo_guess":
100, "duration_sec": 30.0 }) return JSONResponse(out)
app.mount("/media/in", StaticFiles(directory=str(IN_DIR)),
name="media_in") app.mount("/media/out",
StaticFiles(directory=str(OUT_DIR)), name="media_out")
```

5) Flutter — pages minimales (Home + Previews + Paywall)

```
// lib/core/theme.dart import 'package:flutter/material.dart'; final
darkTheme = ThemeData( brightness: Brightness.dark, colorScheme:
ColorScheme.fromSeed(seedColor: Color(0xFF2AE6BE), brightness:
Brightness.dark), scaffoldBackgroundColor: Color(0xFF0B0F10),
useMaterial3: true, );
```

```
// lib/presentation/pages/home.dart (POC) import
'package:flutter/material.dart'; import 'package:dio/dio.dart'; import
'dart:io'; import 'package:record/record.dart'; class HomePage extends
StatefulWidget { const HomePage({super.key}); @override State
createState()=>_HomePageState(); } class _HomePageState extends State {
bool loading = false; final dio = Dio(BaseOptions(baseUrl: const
String.fromEnvironment('BACKEND_BASE'))); final record = AudioRecorder();
Future _captureAndProcess() async { setState(()=>loading=true); final has
= await record.hasPermission(); if(!has){ setState(()=>loading=false);
return; } final path = "/sdcard/Download/shaza_tmp.m4a"; await
record.start(const RecordConfig(encoder: AudioEncoder.aacLc), path: path);
await Future.delayed(const Duration(seconds: 8)); final recPath = await
record.stop(); final file = File(recPath ?? path); final form =
FormData.fromMap({"file": await MultipartFile.fromFile(file.path)}); final
resp = await dio.post("/process", data: form); if(context.mounted)
```

```
Navigator.of(context).push(MaterialPageRoute(builder:
(_)=>PreviewsPage(items: List.from(resp.data))));
setState(()=>loading=false); } @override Widget build(BuildContext
context) { final c = Theme.of(context).colorScheme.primary; return
Scaffold( body: Center( child: GestureDetector( onTap: loading? null :
_captureAndProcess, child: Container( width: 220, height: 220, decoration:
BoxDecoration( shape: BoxShape.circle, gradient: LinearGradient(colors:
[c, c.withOpacity(0.6)]), boxShadow: [BoxShadow(color:
c.withOpacity(0.45), blurRadius: 30, spreadRadius: 4)], ), child:
Icon(loading? Icons.hourglass_bottom : Icons.mic_rounded, size: 88), ), ),
), ); } } // lib/presentation/pages/previews.dart (POC) import
'package:flutter/material.dart'; import
'package:video_player/video_player.dart'; class PreviewsPage extends
StatelessWidget { final List items; const PreviewsPage({super.key,
required this.items}); @override Widget build(BuildContext context) {
return Scaffold( appBar: AppBar(title: const Text('Aperçus 16 s (4
niveaux)')), body: GridView.count( crossAxisCount: 2, childAspectRatio:
16/9, padding: const EdgeInsets.all(12), crossAxisSpacing: 12,
mainAxisSpacing: 12, children: items.map((m)=>_Tile(url: m['preview_url'],
title: "Level ${m['level']}")).toList(), ), bottomNavigationBar: Padding(
padding: const EdgeInsets.all(12), child: ElevatedButton(onPressed: (){
Navigator.of(context).push(MaterialPageRoute(builder: (_)=> const
PaywallPage())); }, child: const Text('Débloquer les 4 pour 1 $')), ), ); }
} class _Tile extends StatefulWidget { final String url; final String
title; const _Tile({required this.url, required this.title}); @override
State<_Tile> createState()=>_TileState(); } class _TileState extends
State<_Tile> { late VideoPlayerController ctrl; @override void
initState(){ super.initState(); ctrl = VideoPlayerController.networkUrl(Ur
i.parse(widget.url))..initialize().then((_){ setState((){}); ctrl.play();
ctrl.setLooping(true); }); } @override void dispose(){ ctrl.dispose();
super.dispose(); } @override Widget build(BuildContext context){ return
Stack(children:[ Positioned.fill(child: ctrl.value.isInitialized ?
VideoPlayer(ctrl) : const Center(child:CircularProgressIndicator())),
Positioned(left:8,top:8, child: Container(color: Colors.black54, padding:
const EdgeInsets.all(6), child: Text(widget.title))),
Positioned(right:8,top:8, child: Container(color: Colors.black87, padding:
const EdgeInsets.all(6), child: const Text("PREVIEW 16s"))), ]); } } //
lib/presentation/pages/paywall.dart (POC — UI seulement) import
'package:flutter/material.dart'; class PaywallPage extends StatelessWidget
{ const PaywallPage({super.key}); @override Widget build(BuildContext
context){ return Scaffold( appBar: AppBar(title: const Text('Débloquer (1
$)')), body: Padding( padding: const EdgeInsets.all(16), child:
Column(crossAxisAlignment: CrossAxisAlignment.stretch, children: [ const
Text("Accès complet aux 4 vidéos (tous les niveaux) — achat unique 1 $."),
const SizedBox(height: 16), ElevatedButton(onPressed: (){/* TODO brancher
in_app_purchase */}, child: const Text("Acheter maintenant (1 $)")),
TextButton(onPressed: (){/* TODO restore */}, child: const Text("Restaurer
l'achat")), ]), ), ); } }
```

6) Firebase & IAP (ultra simple)

• Firebase: crée le projet, active Auth anonyme + Firestore; ajoute ton app Android; place google-services.json dans app/android/app/.

• IAP: crée un produit non■consommable unique “piano_all_levels_1usd”. Dans l’app, quand l’achat réussit → sauvegarde un booléen local (SharedPreferences) + Firestore (users/{uid}/entitlements.allLevels=true).
• Pour la démo immédiate: tu peux laisser le bouton “Acheter” inactif et tester tout le reste.

7) Test rapide (QA) — ce que tu dois voir

- Tap sur le gros bouton: enregistre ~8 s, envoie au backend, puis tu vois une grille 2×2 de vidéos PREVIEW 16 s.
- Chaque tuile lit une vidéo (fond noir POC).
- Le bouton “Débloquer 1 $” ouvre une page paywall (UI, sans achat pour l’instant).
- Le backend répond sur http://10.0.2.2:8000 depuis l’émulateur Android.

Quand tout ça marche, je brancherai pour toi: vrai rendu clavier, IAP Android, Practice Mode, et audio si tu veux.