ShazaPiano — Roadmap & Architecture (End■to■End)

Flutter + FastAPI + Firebase + IAP + CI/CD MVP → 4 Niveaux de Vidéos Piano (1 $ achat unique)

Auteur: Ludo & Assistant (architecture senior)

Date: 31 Octobre 2025

1) Vision & Périmètre

1. Produit: enregistrer ~8 s → extraire la mélodie → générer 4 vidéos de clavier (Lvl1..Lvl4) avec touches animées.
2. Monétisation: achat unique 1 $ (non■consommable) → tous les niveaux à vie.
3. Audio: optionnel (synthèse MIDI avec SoundFont .sf2).
4. Plateforme cible: Android d'abord; iOS ensuite.
5. Backend: FastAPI + FFmpeg + BasicPitch + MoviePy (+ Fluidsynth si audio).
6. Stockage: fichiers en local (MVP), migrables S3/CDN plus tard.

2) Roadmap (jalons)

M1 — MVP (Lvl1 vidéo muette)

- Flutter: enregistrement + upload + lecteur vidéo, gros bouton central.
- Backend: /process?level=1 → m4a→wav→BasicPitch→MIDI→rendu clavier→MP4 (sans audio).
- Firebase: Auth anonyme + Firestore (profil user).
- Telemetry: Crashlytics/Analytics ou Sentry (minimale).
- Critères: MP4 jouable < 60 s; gestion erreurs micro/réseau.

M2 — 4 niveaux + UI Modes

- Flutter: écran Modes (Lvl1..Lvl4), sélection et POST /process?level=.
- Backend: arrange(level) → accompagnements (basse, triades, arpèges).
- Historique local: dernière vidéo + métadonnées.

M3 — Paywall 1 $ (achat unique)

- IAP Android (in_app_purchase): produit non■consommable unique.
- Entitlement persistant: Firestore users/{uid}/entitlements.allLevels=true.
- Restore purchases au démarrage.

M4 — Audio optionnel + robustesse

- Synthèse audio piano via Fluidsynth + .sf2; mux dans MP4 sinon muet.
- Filesystem: purge, quotas, timeouts; règles Firebase strictes.

M5 — CI/CD & Release Alpha

- GitHub Actions: tests + lint + build APK + Docker backend.
- Déploiement backend (Fly.io/Railway/VPS).
- Release Play Console (closed testing).

3) Architecture (monorepo & dossiers)

Monorepo

```
/shazapiano/
■■ app/ # Flutter
■■ backend/ # FastAPI
■■ infra/ # Docker, Nginx, Terraform (later)
■■ docs/ # Specs, checklists
```

Flutter (Clean Architecture)

```
app/lib/
core/ (config, errors, utils)
data/ datasources/ models/ repositories/
domain/ entities/ repositories/ usecases/
presentation/ state/ pages/ widgets/
```

- State: Riverpod • Routing: go_router • HTTP: Dio • Record: record • Video: video_player • IAP: in_app_purchase
- Auth/DB: firebase_core, firebase_auth, cloud_firestore • Prefs: shared_preferences
- Flavors: dev/prod (dart■define BACKEND_BASE, etc.)

Backend (FastAPI)

```
backend/
app.py # routes, static, CORS
inference.py # ffmpeg_to_wav, basicpitch_predict
arranger.py # arrange(level) → PrettyMIDI
render.py # clavier + mux audio
config.py # LEVELS presets, paths, limits
media/in|out/ # I/O • assets/piano.sf2 (option)
```

4) Firebase — création & configuration

- Créer projet (console) → activer Authentication (Anonymous).
- Créer Firestore (mode production, région proche).
- Ajouter app Android (package com.ludo.shazapiano) + google-services.json.
- (Option) Storage si hébergement MP4 via Firebase Storage (MVP: static backend).

Règles Firestore (sécurisées, MVP)

```
rules_version = '2'; service cloud.firestore { match
/databases/{database}/documents { function isSignedIn() { return
request.auth != null; } match /users/{uid} { allow read, write: if
isSignedIn() && uid == request.auth.uid; match /entitlements/{doc} { allow
```

```
read, write: if isSignedIn() && uid == request.auth.uid; } match
/history/{doc} { allow read, write: if isSignedIn() && uid ==
request.auth.uid; } } match /{document=**} { allow read, write: if false; }
} }
```

Modèle de données

```
users/{uid}
entitlements: { allLevels: true|false, purchasedAt }
history/{videoId}: { level, durationSec, keyGuess, createdAt, videoUrl }
```

5) API contrat

POST /process?level=1|2|3|4 (multipart: file=audio)

Réponse 200

```
{ "video_url": "/media/out/abcd_4.mp4", "level": 4, "duration_sec": 27.2,
"key_guess": "G", "tempo_guess": 96 }
```

Erreurs : 400 (bad mime), 413 (too large), 429 (rate limit), 500 (ffmpeg|melody|render).

6) Arrangements — Presets des 4 niveaux

Paramètre	Lvl1 Hyper	Lvl2 Facile	Lvl3 Moyen	Lvl4 Pro
Transpose	→ C maj	→ C maj	Garde tonalité	Originale
Quantification	1/4	1/8	1/8 (+1/16)	1/16
Mélodie	Oui	Oui	Oui	Oui
Main gauche	—	Fondamentale tenue	Fondamentale/Quinte	Arpège 1■5■8
Main droite	—	—	Triade plaquée	Triade brisée
Polyphonie	Non	Faible	Moyenne	Oui
Tempo rendu	0.8×	0.9×	1.0×	1.0× (opt speed)
Plage clavier	C4–G5	C3–C5	C2–C6	C2–C6

7) Sécurité & Performance (backend)

- Rate limit via Nginx: ~20 req/min/IP.
- Max upload: 10 MB (Nginx + validation FastAPI).
- Timeouts: FFmpeg 15 s; BasicPitch 10 s; Rendu 20 s.
- Purge cron: /media/in > 1 j ; /media/out > 7 j.
- Warm■up modèle BasicPitch au boot; jobs parallèles limités (semaphore).
- Logs JSON + /health pour readiness.

8) In■App Purchase (Android)

- Créer produit non■consommable unique: piano_all_levels_1usd.
- in_app_purchase: query product → buy → acknowledge → set entitlements.

- Persistance double: SharedPreferences + Firestore.
- Restore au démarrage (obligatoire iOS, recommandé Android).

9) CI/CD & Déploiement

- GitHub Actions: flutter analyze/test, build APK/AAB, Docker backend.
- Déploiement backend: Fly.io / Railway / VPS (Docker).
- AAB signé → Play Console (closed testing).

10) Release Checklist (Android)

- Créer app Play Console (com.ludo.shazapiano).
- Configurer IAP; comptes test.
- Signer AAB (keystore prod).
- Descriptions, captures, privacy policy.
- Closed track → Open beta → Production.

11) Scripts utiles

```
# Backend cd backend && python -m venv .venv && source .venv/bin/activate
&& pip install -r requirements.txt uvicorn app:app --reload --host 0.0.0.0
--port 8000 # Flutter (émulateur Android) cd app && flutter pub get flutter
run --flavor dev --dart-define=BACKEND_BASE=http://10.0.2.2:8000
```

12) Risques & Parades

- Mélodie bruitée: filtrer notes < 80 ms + quantification douce.
- Rendu lent: 1280×360@30fps, pas d'effets lourds, limiter jobs.
- IAP capricieux: sandbox + restore + logs d’achat.

13) Prochaines Actions (exécutables)

- Initialiser monorepo + dossiers.
- Backend M1: endpoint /process?level=1 (vidéo muette).
- Flutter M1: record → upload → lecture vidéo.
- Firebase: projet + Auth anonyme + Firestore + règles.
- M2: arranger (4 niveaux) + UI Modes.
- M3: IAP 1 $ + entitlements Firestore.
- M4: audio optionnel .sf2 + purges/limits.
- M5: CI/CD + closed testing.

Notes

• Ce document constitue la référence d’architecture initiale. Toute divergence en code doit être synchronisée ici. • Les niveaux sont paramétrables dans config.py (LEVELS).