ARBOT

RUNBOOK COMPLET, SÉQUENCÉ ET OPÉRATIONNEL

0. PRÉ-REQUIS

DESCRIPTIF

Objectif: disposer d'un espace public et stable pour images et docs, et d'outils pour générer index, previews, page, manifest, et automatisations.

ACTIONS

Usager:

1. Créer le dépôt public `Tazevil/Arbot-MiniDB` (branche `main`).
2. Activer Pages: Settings → Pages → Source = `main` / `root`.
3. Installer Python 3.11+ localement si usage local.

Machine:

aucune.

OUTILS/FICHIERS

- GitHub (repo public, Pages).
- Python 3.11+ (local ou Codespaces).

LIVRABLES ATTENDUS

- Dépôt public vide prêt à recevoir `images/`, `docs/`, `json/`, scripts.

ERREURS À ÉVITER

- Dépôt privé.
- Mauvaise casse dans le nom du dépôt lors des URLs.

POINTS IMPORTANTS

- Tout est public. Nettoyer les EXIF (GPS).
- Respecter la casse exacte: `Arbot-MiniDB`.

AUDIT

- Repo public ✔
- Pages activées ✔

INCERTITUDES

- Aucune.

SOURCES

UNKNOWN

ARBOT

RUNBOOK COMPLET, SÉQUENCÉ ET OPÉRATIONNEL

1. STRUCTURATION DES IMAGES + NOMMAGE

DESCRIPTIF

Objectif: fiabiliser la lecture automatique par regex A/B/C, décoder `Id_File`, classer “chantier” vs “sinistre”.

ACTIONS

- Usager:
  1. Placer les images dans `images/**` selon ta taxonomie.
  2. Respecter les patterns:
     - Cas A (id_zone=0 chantier): `^(\d{4})_DETAIL_SPEC_(\d{8})\.(ext)$`
     - Cas B (id_zone=1 plan): `^(\d{4})_DETAIL_(\d{4})\.(ext)$`
     - Cas C (id_zone ∈ {2..9}): `^(\d{4})_DETAIL_SPEC\.(ext)$`
- Machine: aucune.

OUTILS/FICHIERS

- Dictionnaires zones/cats/specs (ceux que tu as fournis).

LIVRABLES ATTENDUS

- `images/**` remplis avec fichiers nommés selon A/B/C.

ERREURS À ÉVITER

- Espace dans `detail`.
- Spéc non listée.
- Date absente en Cas A.
- `Id_File` non cohérent.

POINTS IMPORTANTS

- Classification: “chantier” si date AAAAMMJJ juste avant l’extension, sinon “sinistre”.
- Unicité `id_image` par `(Id_Zone, Id_Cat)`.

AUDIT

- Vérifier sur un échantillon: `Id_File == Id_Zone*1000 + Id_Cat*100 + id_image` ✔

INCERTITUDES

- Mapping numérique `Id_Spec`: si requis, fournir la table.

SOURCES

UNKNOWN

2. PRÉVIEWS WEBP + ORIENTATION EXIF + CHAMPS ENRICHIS

DESCRIPTIF

Objectif: accélérer la vision et normaliser l'orientation, tout en ajoutant `url_preview` et `roi_hints` par défaut à l'index

ACTIONS

- Usager: placer le script à la racine sous `make_previews_and_augment_json.py`, puis exécuter.
- Machine: crée `images/preview/**.webp`, met à jour `json/images_db.json`.

OUTILS/FICHIERS

- `make_previews_and_augment_json.py` (fourni précédemment).
- Pillow: `pip install pillow`.

COMMANDE

```
python make_previews_and_augment_json.py ^
 --images .\images ^
 --out_json .\json\images_db.json ^
 --owner Tazevil --repo Arbot-MiniDB --branch main
```

LIVRABLES ATTENDUS

- `images/preview/**.webp`
- `json/images_db.json` enrichi (`url_preview`, `roi_hints`, `phase`)

ERREURS À ÉVITER

- Previews écrasent les originaux: non, le script ne le fait pas.
- Manque de dépendance Pillow.

POINTS IMPORTANTS

- Côté long cible 1280–2048 px.
- `roi_hints` centrés par défaut, ajustables plus tard.

AUDIT

- Previews existants ✔
- `images_db.json` mis à jour ✔

INCERTITUDES

- Taille optimale = dépend du réseau et du LLM.

3. VALIDATION DES NOMS + GÉNÉRATION SIDECAR CSV ET INDEX IMAGES

DESCRIPTIF

Objectif: valider regex A/B/C, unicité, cohérence `Id_File`, et produire `images_sidecar.csv` + `images_db.json`.

ACTIONS

- Usager: exécuter le validateur.
- Machine: parse récursif `images/**` (version patchée `rglob`), génère CSV + JSON.

OUTILS/FICHIERS

- `validate_pack.py` (version patchée `rglob` pour sous-dossiers).

COMMANDE

```
python validate_pack.py --images .\images --out .\json ^
 --owner Tazevil --repo Arbot-MiniDB --branch main
```

LIVRABLES ATTENDUS

- `json/images_sidecar.csv`
- `json/images_db.json` (si non présent, créé. Sinon tu peux fusionner avec la version augmentée de l'étape 2).

ERREURS À ÉVITER

- `fatal: not a git repository`: ignorable si tu ne `git push` pas localement.
- `utcnow()` déprécié: utiliser `now(timezone.utc)`.

POINTS IMPORTANTS

- Classification "chantier" si date en fin de nom (Cas A).
- `Id_File` unique.

AUDIT

- Aucune ligne "Duplicat" ou "Incohérence Id_File" ✔

INCERTITUDES

- Fusion éventuelle avec le JSON enrichi: arbitrer l'ordre (valider puis enrichir, ou l'inverse).

SOURCES

UNKNOWN

4. INDEX DES DOCUMENTS (DTU, NOTICES) POUR CITATIONS

DESCRIPTIF

Objectif: exposer une liste publique de PDF pour permettre les citations `doc_id + page`.

ACTIONS

- Usager: placer PDF en `docs/`; créer `scripts/generate_docs_index.py` (fourni) ou composer `json/docs_index.json` à la main.
- Machine: génère `json/docs_index.json`.

OUTILS/FICHIERS

- `scripts/generate_docs_index.py` (fourni).

COMMANDE

python .\scripts\generate_docs_index.py

LIVRABLES ATTENDUS

- `json/docs_index.json` avec `{id,title,type,url}`.

ERREURS À ÉVITER

- URL privées/non accessibles.
- PDF scanné sans couche texte: citations pages restent `UNKNOWN`.

POINTS IMPORTANTS

- `doc_id` stable et court.
- Titres explicites.

AUDIT

- URLs RAW ouvrables ✔

INCERTITUDE

- Texte extractible vs PDF image.

SOURCES

UNKNOWN

5. GÉNÉRATION DE LA GALERIE INDEX.HTML

DESCRIPTIF

Objectif: page publique récapitulative des images pour contrôle visuel

ACTIONS

- Usager: déposer `generate_index_html.py` à la racine, exécuter.
- Machine: crée `index.html`.

OUTILS/FICHIERS

- `generate_index_html.py` (fourni, récursif).

COMMANDE

python generate_index_html.py --images .\images --out .\index.html

LIVRABLES ATTENDUS

- `index.html` à la racine.

ERREURS À ÉVITER

- Chemins absolus dans HTML: utiliser chemins relatifs (`images/...`).

POINTS IMPORTANTS

- Pages: main/root activé.
- Casse correcte.

AUDIT

- `https://tazevil.github.io/Arbot-MiniDB/` s'affiche ✔

INCERTITUDES

- Aucune.

SOURCES

UNKNOWN

6. CROPS DEPUIS ROI ET AJOUT URL_CROP (OPTION)

DESCRIPTIF

Objectif: fournir des vues serrées sur défauts pour améliorer la lecture VISION.

ACTIONS

- Usager: exécuter le script `make_crops_from_roi.py`.
- Machine: crée `images/crop/*_crop.jpg`, ajoute `url_crop` dans `images_db.json`.

OUTILS/FICHIERS

- `make_crops_from_roi.py` (fourni).

COMMANDE

python make_crops_from_roi.py

LIVRABLES ATTENDUS

- `images/crop/*_crop.jpg`
- `json/images_db.json` mis à jour (`url_crop`)

ERREURS À ÉVITER

- ROI hors image.
- Absence de `roi_hints`: le script a un défaut centré.

POINTS IMPORTANTS

- Garder précision des noms.

AUDIT

- `url_crop` ouverts en RAW ✔

INCERTITUDES

- Taille ROI par défaut à ajuster selon cas.

SOURCES

UNKNOWN

7. TUILAGE 1024 PX + 10% RECOUVREMENT (OPTION)

DESCRIPTIF

Objectif: découper grandes images en patchs pour petits défauts.

ACTIONS

- Usager: exécuter `tile_1024_overlap.py`.
- Machine: produit `tiles/**` et `tiles_map.json`.

OUTILS/FICHIERS

- `tile_1024_overlap.py` (fourni).

COMMANDE

python tile_1024_overlap.py --src .\images --out .\tiles --size 1024 --overlap 0.10

LIVRABLES ATTENDUS

- `tiles/*.jpg`
- `tiles/tiles_map.json`

ERREURS À ÉVITER

- Oublier de relier patch → image source dans l'analyse.

POINTS IMPORTANTS

- Analyser d'abord les previews/crops avant d'augmenter le volume.

AUDIT

- `tiles_map.json` cohérent ✔

INCERTITUDES

- Charge API vs gain de détection.

SOURCES

UNKNOWN

8. MANIFESTES SHA-256 (TRAÇABILITÉ)

DESCRIPTIF

Objectif: prouver l'intégrité des fichiers.

ACTIONS

- Usager: exécuter `generate_checksum.bat` et/ou `build_manifest.py`.
- Machine: produit `checksums_sha256.txt` et `manifest.json` avec `global_sha256`.

OUTILS/FICHIERS

- `generate_checksum.bat` (batch glob).
- `build_manifest.py` (JSON + hash global, fourni).

COMMANDES

```
scripts\generate_checksum.bat
python build_manifest.py
```

LIVRABLES ATTENDUS

- `checksums_sha256.txt`
- `manifest.json` (avec `global_sha256`)

ERREURS À ÉVITER

- Inclure `.git/` dans le manifest: script l'exclut.

POINTS IMPORTANTS

- Hasher après gel des fichiers.

AUDIT

- Hash global stable entre deux pulls identiques

INCERTITUDES

- Aucune.

SOURCES

UNKNOWN

9. PUBLICATION SUR GITHUB

DESCRIPTIF

Objectif: rendre tout accessible au LLM et au navigateur.

ACTIONS

Usager (au choix):
UI GitHub: Add file → Upload files pour `images/**`, `json/**`, `docs/**`, `index.html`.
Codespaces: ouvrir un terminal, exécuter scripts, commit/push.
Git local:
git init
git add .
git commit -m "feat: previews, indexes, page"
git branch -M main
git remote add origin https://github.com/Tazevil/Arbot-MiniDB.git
git push -u origin main

-
- Machine: héberge, met en cache CDN.

OUTILS/FICHIERS

- GitHub Pages actif.

LIVRABLES ATTENDUS

- URLs RAW valides.
- Page `index.html` en ligne.

ERREURS À ÉVITER

- Casse des chemins.
- Delai de cache CDN.

POINTS IMPORTANTS

- Tester une URL RAW d'image au hasard.

AUDIT

- `raw.githubusercontent.com/.../images/...` s'ouvre ✔
- `…/json/images_db.json` s'ouvre ✔

INCERTITUDES

- Latence CDN 1–10 min.

SOURCES

UNKNOWN

10. AUTOMATISATION GITHUB ACTIONS (OPTION)

DESCRIPTIF

Objectif: tout régénérer à chaque ajout d'images

ACTIONS

- Usager: créer `.github/workflows/build-all.yml` (fourni précédemment).
- Machine: exécute previews → validate → index.html → commit.

OUTILS/FICHIERS

- `build-all.yml` (fourni).

LIVRABLES ATTENDUS

- Commits automatiques: previews, JSON, CSV, page.

ERREURS À ÉVITER

- Permissions `contents: write` manquantes.

POINTS IMPORTANTS

- Déclencheurs sur `images/**` et scripts.

AUDIT

- Job "build" vert dans l'onglet Actions ✔

INCERTITUDES

- Aucune.

SOURCES

UNKNOWN

11. PROMPTS VISION GPT-4O: PASSES ET LOTISATION

DESCRIPTIF

Objectif: analyser par lots avec contrôle et citations DTU.

ACTIONS

- Usager: coller les prompts fournis (Pass 0 CSV, Pass 1 JSON + citations, Pass 2 SVG).
- Machine: charge les URLs, exécute la vision, renvoie les formats stricts.

OUTILS/FICHIERS

- Prompts fournis plus haut.
- `json/images_db.json`, `json/docs_index.json`.

LIVRABLES ATTENDUS

- CSV de pré-passe (contexte).
- JSON d'analyse avec citations.
- SVG d'overlays.

ERREURS À ÉVITER

- Température > 0.2.
- Absence de `docs_index.json` si citations requises.

POINTS IMPORTANTS

- Commandes "SUIVANT/STOP" pour lotiser.
- Few-shot: exemple mauvais vs bon.

AUDIT

- Sorties strictes sans narration ✔

INCERTITUDES

- Pages exactes dans PDF scannés.

SOURCES

UNKNOWN

12. QUALITÉ DE RENDU ET CONTRÔLES FINAUX

DESCRIPTIF

Objectif: garantir lisibilité et cohérence.

ACTIONS

- Usager: vérifier 5–10 images échantillons, corriger `roi_hints`, relancer Pass 1 si besoin.
- Machine: aucune.

OUTILS/FICHIERS

- `images_db.json` éditable.
- Scripts déjà listés.

LIVRABLES ATTENDUS

- Résultats homogènes, citations présentes, `confidence` renseigné.

ERREURS À ÉVITER

- Confondre hypothèse et constat visuel.
- Oublier `UNKNOWN` quand doute.

POINTS IMPORTANTS

- Ajouter `confidence ∈ [0,1]` et `incertitude` par défaut.
- Traçabilité: manifest + hashes conservés au repo.

AUDIT

- 0 lien cassé, 0 image manquante, 0 règle regex violée ✔

INCERTITUDES

- Aucune.

SOURCES

UNKNOWN

13. RÉCAPITULATIF FICHIERS CLÉS À LA RACINE

- `validate_pack.py`
- `make_previews_and_augment_json.py`
- `generate_index_html.py`
- `make_crops_from_roi.py`
- `tile_1024_overlap.py`
- `build_manifest.py`
- `scripts/generate_checksum.bat`
- `scripts/generate_docs_index.py`
- `.github/workflows/build-all.yml`
- `json/images_db.json`, `json/docs_index.json`
- `index.html`
- `images/**`, `docs/**`