**Checkliste: Post-Patch-Triage (immer gleiche Reihenfolge)**

**Ziel:** Nach Vibe/Patch systematisch zu grün.

1. **Import/Config zuerst**
2. # häufigster Crash: Config validiert beim Import
3. Test-Path .\config\pald\_schema.json # muss True sein

Falls nicht: Datei an Ort kopieren (haben wir erledigt).

1. **Syntaxfehler in Tests/Code**
2. python -m py\_compile tests\\*.py src\\*\*\\*.py # grober Syntax-Scan

→ Zeigt dir genau Datei+Zeile.

1. **Unit-Tests iterativ**
2. pytest -q --maxfail=1 --disable-warnings

→ Den **ersten** Fehler fixen, wiederholen.

1. **Migrations/Models konsistent**
   * Stimmen Tabellennamen/Feldnamen zwischen models.py und migrations?
   * Einmal alembic upgrade head (falls du Alembic nutzt) – in Test-DB oder SQLite-tmp.
2. **Laufzeit-Kontrakte der UI**
   * Wenn UI-Tests brechen: Dataclass/Response-Contracts prüfen (z. B. pald\_light, pald\_diff\_summary, defer\_notice vorhanden?).
3. **Quality Hooks (optional)**
4. # falls vorhanden
5. pre-commit run --all-files

**4) Checkliste: Vibe → Patch → Apply (kostenarm & robust)**

**Vorbereitung**

# Branch-Hygiene

git switch pald\_enhancement

git switch -c feat/pald\_enhancement-<bundle>

mkdir patches

git status

**Vibe-Prompt (Beispiel)**

We are in Vibe. Generate a **unified diff** (git-compatible) for the following files: …  
**Do not execute anything.** Output **only the patch** in one code block. Keep exact paths.

**Patch speichern & sanitisieren**

# Patch speichern

Set-Content .\patches\bundle.patch -Encoding utf8 -Value @'

<PASTE VIBE PATCH 1:1 HIER REIN>

'@

# Sanitize: Kopfzeilen und Codefences entfernen, EOL normalisieren

$raw = Get-Content .\patches\bundle.patch -Raw -Encoding utf8

$raw = $raw -replace '(?m)^index\s+.\*\r?\n',''

$raw = $raw -replace '(?m)^```.\*\r?\n',''

$raw = $raw -replace '\r',''

Set-Content .\patches\bundle.patch -Value $raw -Encoding utf8

**Trockenlauf, dann anwenden**

git apply --check .\patches\bundle.patch \*> $null

if ($LASTEXITCODE -eq 0) { git apply .\patches\bundle.patch } else { Write-Host "Patch needs split/reject." }

**Wenn „corrupt/does not apply“ → Split & Reject**

# Splitten pro Datei

$lines = Get-Content .\patches\bundle.patch -Encoding utf8

$starts=@(); for($i=0;$i -lt $lines.Count;$i++){ if($lines[$i] -match '^diff --git ') {$starts+=$i} }

for($b=0;$b -lt $starts.Count;$b++){

$start=$starts[$b]; if($b -lt $starts.Count-1){$end=$starts[$b+1]-1}else{$end=$lines.Count-1}

$chunk=$lines[$start..$end]

$name="chunk\_$b"; if($chunk[0] -match '^diff --git a/(.+?) b/'){$name=($Matches[1] -replace '[\\/: ]','\_')}

$out=(".\patches\split\_{0:d2}\_{1}.patch" -f $b,$name)

[IO.File]::WriteAllLines($out,$chunk,(New-Object System.Text.UTF8Encoding($false)))

}

# Gute Splits direkt anwenden, schlechte im Reject-Modus

Get-ChildItem .\patches\split\_\*.patch | ForEach-Object {

git apply --check $\_.FullName \*> $null

if ($LASTEXITCODE -eq 0) { git apply $\_.FullName }

else { git apply --reject --whitespace=nowarn $\_.FullName }

}

# Danach .rej Dateien manuell mergen (VS Code), committen

Get-ChildItem -Recurse -Filter \*.rej

**Tests, Commit, Push**

pytest -q

git add -A

git commit -m "feat(bundle): apply Vibe patch via split/reject"

git push -u origin feat/pald\_enhancement-<bundle>

**PR → Integrations-Branch**

* GitHub PR: Base = pald\_enhancement, Compare = feat/pald\_enhancement-<bundle>.

**5) Git-Rettungsanker (du wirst sie nutzen)**

# Zurück zum letzten Commit (nur im Notfall, verliert uncommitted):

git reset --hard HEAD

# Letzten Commit rückgängig machen, Änderungen behalten:

git reset --soft HEAD~1

# Einzelne Datei vom letzten Commit zurückholen:

git restore --source=HEAD -- path\to\file

# Branch aus Remote neu anlegen:

git switch -c mybranch origin/mybranch

# Working changes parken:

git stash push -u -m "WIP"

git stash apply stash@{0}