

委譲判定シート

AI の指摘を4つに仕分けます。ハンズオン1とハンズオン2で使います。

AI が出した指摘を、4分類のどれに入れるかを決めるシートです。ハンズオン1(Issue #1)とハンズオン2(Issue #2・#3)で使います。

指摘を全部直すと、5行の修正に半日かかります。全部見送ると、レビューを回す意味がなくなります。この2つの間に線を引くために、指摘ごとに3軸を先に決めてから分類します。分類から入ると、その日の気分で線が動きます。

3軸

軸	値	見るところ
可逆性	戻せる / 戻しにくい / 戻せない	コードだけの変更は戻せます。DB のスキーマとデータ、外部連携、権限の変更は戻せません。コードを <code>revert</code> してもデータは戻りません
機械検証	CI で落ちる / テストを足せば落ちる / 人が読むしかない	今の CI(lint 8.3、lint 5.4、DB テスト、AI レビューのゲート)で落ちるかどうかです。「落ちるはず」ではなく、落ちたのを見たかどうかで書きます
影響範囲	1ファイル内 / 同一機能 / 共通処理と複数機能	<code>app/libraries/</code> と <code>app/core/</code> に触る指摘は、常に一番右です

4分類の決め方

3軸を書いたから、次の順で当てはめます。上から順に見て、最初に当たったところで止めます。

- 可逆性が「戻せない」なら、機械検証の値に関わらず **要調査**。人が読んでから決めます
- 機械検証が「CI で落ちる」なら **即修正**。人が採否を議論する対象ではありません。議論した時点で工数が二重になります
- 影響範囲が「共通処理と複数機能」で、可逆性が「戻しにくい」なら **将来課題**。この MR では直しません
- 指摘の前提がこのリポジトリの設計と食い違っているなら **意図した設計**。理由を1行書いて `known_issues.md` に K 番号で登録します
- 残りは **即修正** か **将来課題** を、この MR の目的に照らして選びます

「意図した設計」に入れたものは、次から同じ指摘が出ないように記録するところまでが1セットです。記録しないと、同じ判断を毎回の改修で繰り返します。小さな改修ほど指摘対応に時間がかかる原因の一つがここにあります。

記入欄

N o	出どころ	指摘の要約	可 逆 性	機械検証	影響 範囲	分 類	次の一手	記 入 者
1	同一セッション	estimate_ctl.php の customer_name が SQL 文字列に連結されていて、' OR '1'='1' で全件返る	戻せる	テストを足せば落ちる	同一機能	即修正	db_select の第2引数へブレースホルダで渡し、検索のテストを1本足す	
2								
3								
4								
5								
6								
7								
8								
9								
10								

出どころは「同一セッション」「別セッション」「Codex」「CI」のどれかを書きます。同じ指摘が2か所から出たときは、両方に行を立てずに1行にまとめ、出どころを2つ並べてください。重なった数がそのまま、レビューを何本回すかの判断材料になります。

集計欄

項目	件数
指摘の総数	
即修正	
要調査	
将来課題	

項目	件数
意図した設計	
このうち CI で落ちたもの	

一番下の行が大きいほど、人が読む量を減らせます。人が読む量を減らす手は「基準表を厳しくする」ではなく「CIに落とせるものを CI へ移す」です。

