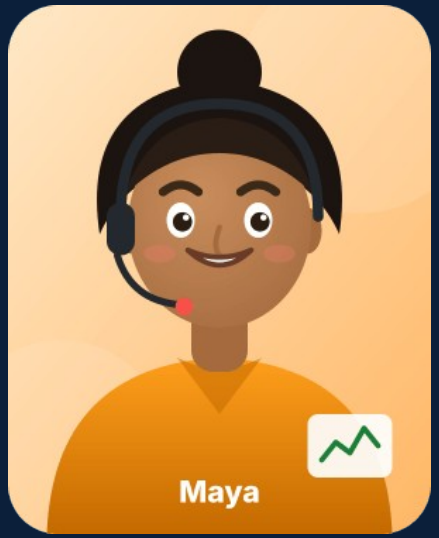
HEF-150 · MISSION 02 / 10

Discovery et problèmes utilisateurs

Les gens mentent sur l'avenir sans le vouloir. Une bonne enquête ne leur demande pas ce qu'ils voudraient : elle observe ce qu'ils ont déjà fait.



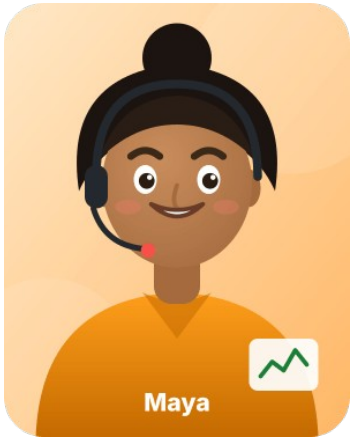


Badge : Enquêteur de besoins

100 min · avancé

L'histoire

Pour savoir si les membres veulent un « mode communauté », Kodi envoie un sondage. 87 % répondent « oui, bonne idée ». Il lance la fonctionnalité : trois semaines plus tard, 4 % des membres l'ont ouverte. Maya revient avec dix entretiens. Elle ne pose aucune question sur le mode communauté. Elle demande : « Racontez-moi la dernière fois que vous êtes resté bloqué sur un cours. Qu'avez-vous fait ? » Sept personnes sur dix répondent : « J'ai cherché sur un moteur de recherche, puis posé la question à un collègue. »



« Le sondage mesurait la politesse, dit Maya. Les entretiens ont mesuré des faits. Le vrai problème n'est pas "pas de communauté" : c'est "être bloqué et ne pas savoir à qui demander au bon moment". La solution peut être autre chose qu'une communauté. »

L'histoire (suite)

Tu apprends à conduire des entretiens utiles (la règle du « Mom Test »), à distinguer faits et opinions, à regrouper des observations en thèmes, à estimer combien d'entretiens suffisent, à évaluer l'importance et la satisfaction des besoins, et à construire un arbre d'opportunités.

Ta mission

À la fin, je serai capable de…

1. conduire un entretien de découverte centré sur le passé et les faits
2. classer des phrases d'entretien en faits, opinions et promesses futures
3. regrouper des observations en thèmes (affinités) et estimer la saturation
4. calculer la marge d'erreur d'un sondage et en reconnaître les biais
5. hiérarchiser des opportunités avec importance et satisfaction, et bâtir un arbre d'opportunités

Défi d'échauffement

Défi d'échauffement

Un ami veut ouvrir un café. Tu lui demandes : « Tu viendrais dans mon café ? » Il répond : « Bien sûr ! » Que vaut cette réponse ? Maintenant tu lui demandes : « La dernière fois que tu as voulu un café avant le travail, où es-tu allé, combien as-tu payé, et qu'est-ce qui t'a agacé ? » Laquelle des deux questions t'apprend le plus, et pourquoi ?

Le concept

La découverte (discovery)

La **découverte produit** est le travail continu qui consiste à comprendre les **problèmes** des utilisateurs **avant** de construire. Elle réduit le risque de **valeur** (mission 1) à moindre coût. Elle est menée par le **trio** produit / design / technique, **chaque semaine**, pas une fois par an.

Les méthodes

Méthode	Apprend	Limite
Entretien (5 à 8 personnes d'un segment)	**pourquoi** et **comment**, contexte, vocabulaire	petits échantillons, pas de chiffres généralisables
Observation (suivre quelqu'un au travail)	ce que les gens **font** réellement	coûteux

Le concept

Les méthodes (suite)

Méthode	Apprend	Limite
Sondage	**combien**, répartition, mesure	mauvais pour découvrir, biais de réponse
Données d'usage	ce que les gens **font** sur le produit	ne dit pas **pourquoi**
Support, avis, tickets	douleurs récurrentes	biais vers les mécontents
Tests d'usage	difficultés avec une interface	pas la valeur du produit

Principe : combiner **qualitatif** (pourquoi) et **quantitatif** (combien).

La règle du « Mom Test »

Une maman dit toujours que ton idée est géniale. Pour obtenir une information fiable :

Le concept

La règle du « Mom Test » (suite)

1. **Parle de leur vie, pas de ton idée.**
2. **Demande des faits passés**, pas des opinions ni des promesses (« comment as-tu fait la dernière fois ? » plutôt que « tu utiliserais ? »).
3. **Parle moins, écoute plus.**
4. **Creuse** : « pourquoi ? », « raconte-moi », « et ensuite ? », « combien ça t'a coûté ? ».
5. **Cherche les engagements** (temps, argent, réputation) plutôt que les compliments.

Phrase entendue	Nature	Valeur
« J'ai essayé trois outils le mois dernier. »	**fait passé**	forte
« Je paie 30 € par mois pour ça. »	**fait / engagement**	très forte
« Ça serait super d'avoir ça. »	opinion / compliment	faible
« Je l'utiliserais sûrement. »	promesse future	quasi nulle

Le concept

La règle du « Mom Test » (suite)

Phrase entendue	Nature	Valeur
« Pourquoi pas un tableau de bord ? »	solution proposée	à creuser : quel problème ?

Biais à connaître

Biais	Effet
Politesse / désirabilité sociale	on répond ce qui plaît
Confirmation	on retient ce qui confirme l'idée
Échantillon biaisé	interroger des amis, des fans, des gens faciles à joindre
Question orientée	« Ne trouvez-vous pas que… ? »
Biais de survie	on n'interroge que ceux qui sont restés

Le concept

Du verbatim aux thèmes

1. Noter des **citations** et des **faits** (pas des interprétations).
2. **Regrouper par affinité** : chaque thème = un type de problème.
3. Compter **combien de personnes** (pas combien de citations) évoquent chaque thème.
4. Distinguer **fréquence** et **intensité** (la douleur, le coût).

Combien d'entretiens ?

Si un problème touche une proportion p des utilisateurs, la probabilité qu'**au moins une** personne sur n l'évoque est $1 - (1 - p)^n$. Un problème qui touche 30 % des gens est entendu au moins une fois dans 5 entretiens avec 83 % de probabilité. On s'arrête à la **saturation** : quand de nouveaux entretiens n'apportent plus de thème nouveau.

Le concept

Sondage : marge d'erreur

Pour une proportion `p` observée sur `n` répondants (confiance 95 %) : marge ≈ `1,96 × √(p(1−p)/n)`. Avec n = 100 et p = 50 % : ± 9,8 points. Avec n = 1 000 : ± 3,1 points. La marge ne protège **pas** des biais d'échantillon.

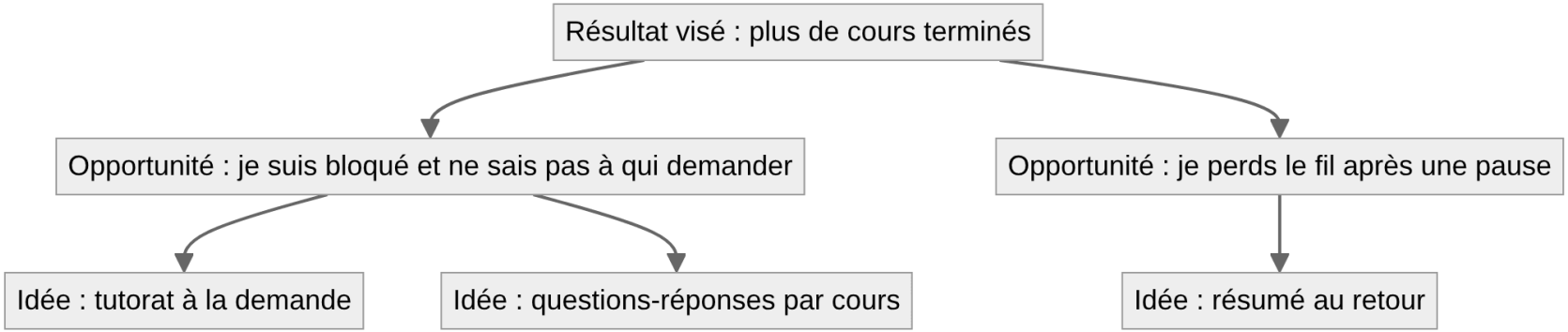
Importance et satisfaction

Pour hiérarchiser des besoins (Ulwick) : demander l'**importance** et la **satisfaction** actuelle (échelle 1 à 10). L'**opportunité** = `importance + max(importance − satisfaction, 0)`. Haute importance + basse satisfaction = **opportunité forte**.

L'arbre d'opportunités

Le concept

L'arbre d'opportunités (suite)



Description du schéma : un résultat visé en racine, deux opportunités (problèmes des utilisateurs) en dessous, et plusieurs idées de solutions sous chaque opportunité.

On choisit une opportunité, on explore **plusieurs** solutions possibles (jamais une seule), et on teste (mission 3).

Le concept

L'arbre d'opportunités (suite)

🧠 ***Astuce*** *: après chaque entretien, écris ce que la personne a* ***fait****, pas ce qu'elle a* ***dit qu'elle ferait****. Si ton compte rendu ne contient que des opinions, l'entretien n'a rien appris.*

Exemple pas à pas

Problème : classer des phrases d'entretien (fait, engagement, opinion, promesse, solution), estimer par simulation combien d'entretiens sont nécessaires, regrouper des observations en thèmes, calculer la marge d'erreur d'un sondage et classer des opportunités selon importance et satisfaction.

```
import math, random, re
from collections import Counter, defaultdict

# ---------------- 1. fait ou opinion ? règle du Mom Test ----------------
def classer(phrase):
    p = phrase.lower()
    if re.search(r"\b(je paie|j'ai payé|j'ai acheté|je paye|abonn)", p): return "e
   ngagement"
    if re.search(r"\b(j'ai|je suis allé|hier|la semaine dernière|le mois dernier|l
   a dernière fois|j'étais)\b", p): return "fait passé"
    if re.search(r"\b(je ferais|j'utiliserais|je voudrais|je serais|ça serait|ce s
   erait|je pourrais|sûrement|probablement)\b", p): return "promesse / opinion"
    if re.search(r"\b(il faudrait|pourquoi pas|vous devriez|et si)\b", p): return
   "solution proposée"
```

Exemple pas à pas

Exemple pas à pas (suite)

```
    return "autre"

NOTES = ["La dernière fois, j'ai cherché sur un moteur de recherche puis j'ai dema
   ndé à un collègue.",
         "Je paie 30 € par mois pour une plateforme de cours.",
         "Ça serait super d'avoir une communauté.",
         "J'utiliserais sûrement un tableau de bord.",
         "Pourquoi pas un forum ?",
         "La semaine dernière, je suis resté bloqué deux heures sur un exercice."]
resultats = [classer(n) for n in NOTES]
for n, r in zip(NOTES, resultats): print(f"  {r:20s} ← {n}")
assert resultats == ["fait passé", "engagement", "promesse / opinion", "promesse /
    opinion", "solution proposée", "fait passé"]
print("part de faits dans ces notes :", f"{sum(r in ('fait passé', 'engagement') f
   or r in resultats) / len(resultats):.0%}")

# ---------------- 2. combien d'entretiens ? ----------------
def proba_au_moins_un(p, n): return 1 - (1 - p) ** n
print("n entretiens | problème touchant 10 % | 30 % | 50 %")
```

Exemple pas à pas

Exemple pas à pas (suite)

```
for n in (3, 5, 8, 12, 20):
    print(f"     {n:2d}      |        {proba_au_moins_un(0.1, n):5.0%}          |
   {proba_au_moins_un(0.3, n):4.0%} | {proba_au_moins_un(0.5, n):4.0%}")
assert round(proba_au_moins_un(0.3, 5), 2) == 0.83 and round(proba_au_moins_un(0.1
   , 20), 2) == 0.88

# saturation des thèmes par simulation : 12 thèmes de prévalences inégales, entret
   iens successifs
rng = random.Random(1)
PREVALENCES = [0.7, 0.5, 0.4, 0.3, 0.25, 0.2, 0.15, 0.12, 0.1, 0.08, 0.05, 0.03]
def decouverts_apres(n, essais=2000):
    total = 0
    for _ in range(essais):
        vus = set()
        for _ in range(n):
            for i, p in enumerate(PREVALENCES):
                if rng.random() < p: vus.add(i)
        total += len(vus)
    return total / essais
```

Exemple pas à pas

Exemple pas à pas (suite)

```
print("entretiens | thèmes découverts (sur 12)")
courbe = {n: decouverts_apres(n) for n in (1, 3, 5, 8, 12, 20, 30)}
for n, v in courbe.items(): print(f"    {n:2d}     |  {v:5.2f}")
assert courbe[5] > 7 and courbe[30] < 12 and courbe[8] - courbe[5] < courbe[5] - c
   ourbe[3] + 1       # rendements décroissants

# ---------------- 3. regroupement par affinité (mots-clés → thèmes) -------------
   ---
THEMES = {"blocage": ("bloqué", "coincé", "perdu", "comprends pas", "bloque"),
          "temps": ("temps", "pas le temps", "fatigué", "soir", "trop long"),
          "motivation": ("motivation", "abandonné", "lâché", "seul", "isolé"),
          "prix": ("cher", "prix", "payer", "abonnement")}
VERBATIMS = ["Je me suis retrouvé bloqué sur la récursivité pendant deux heures",
   "Je n'ai pas le temps en semaine, seulement le soir", "J'ai abandonné au bout d
   e trois semaines, je me sentais seul",
             "Je suis coincé dès qu'il y a une erreur que je ne comprends pas", "C
   'est trop long, je suis fatigué le soir", "L'abonnement est un peu cher pour ce
    que je consomme", "J'ai lâché parce que personne ne répondait à mes questions"
   ]
```

Exemple pas à pas

Exemple pas à pas (suite)

```
par_theme = defaultdict(list)
for i, v in enumerate(VERBATIMS):
    for theme, mots in THEMES.items():
        if any(re.search(r"\b" + re.escape(m) + r"\b", v.lower()) for m in mots):
   par_theme[theme].append(i)
print({t: len(ids) for t, ids in sorted(par_theme.items(), key=lambda kv: -len(kv[
   1]))})
assert len(par_theme["blocage"]) == 2 and len(par_theme["temps"]) == 2 and len(par
   _theme["motivation"]) == 2

# ---------------- 4. marge d'erreur d'un sondage ----------------
def marge(p, n): return 1.96 * math.sqrt(p * (1 - p) / n)
for n in (50, 100, 400, 1000, 2500):
    print(f"  n = {n:4d} : ± {marge(0.5, n) * 100:4.1f} points")
assert round(marge(0.5, 100) * 100, 1) == 9.8 and round(marge(0.5, 1000) * 100, 1)
    == 3.1
n_necessaire = math.ceil((1.96 / 0.03) ** 2 * 0.25)
print("répondants nécessaires pour ± 3 points :", n_necessaire)
assert n_necessaire == 1068
```

Exemple pas à pas (suite)

```
# ---------------- 5. importance, satisfaction, opportunité ----------------
BESOINS = {"trouver de l'aide quand je suis bloqué": (9.0, 3.0), "suivre ma progre
   ssion": (7.0, 6.5), "choisir par quoi commencer": (8.5, 4.5), "personnaliser le
   s couleurs": (3.0, 3.0), "télécharger les cours": (6.0, 7.0)}
def opportunite(importance, satisfaction): return importance + max(importance - sa
   tisfaction, 0)
classement = sorted(BESOINS.items(), key=lambda kv: -opportunite(*kv[1]))
for besoin, (i, s) in classement: print(f"  {opportunite(i, s):5.1f}  importance {
   i} / satisfaction {s}  ← {besoin}")
assert classement[0][0] == "trouver de l'aide quand je suis bloqué" and classement
   [-1][0] in ("personnaliser les couleurs", "télécharger les cours")
```

Lecture :

1. Le classeur applique la règle du Mom Test : seules les phrases au **passé** et les **engagements** sont des faits. Les compliments, promesses et suggestions de solutions sont à manier avec prudence.

Exemple pas à pas (suite)

2. Un problème qui touche 30 % des gens est entendu au moins une fois dans 5 entretiens avec 83 % de probabilité ; les thèmes découverts suivent une courbe à **rendements décroissants** : après 8 à 12 entretiens, on apprend peu de choses nouvelles.
3. Le regroupement par affinité compte les **personnes** par thème : « blocage », « temps », « motivation » ressortent à égalité ici.
4. Pour un sondage, la marge d'erreur passe de ± 9,8 points (n = 100) à ± 3,1 points (n = 1 000) — il faut **1 068 répondants** pour ± 3 points. Mais une marge faible n'élimine pas les biais d'échantillon.
5. Le besoin « trouver de l'aide quand je suis bloqué » (importance forte, satisfaction faible) est la plus forte **opportunité**.

À toi, avec aide

À toi, avec aide

Tu veux savoir si le problème « les nouveaux membres ne savent pas par où commencer » est assez répandu. D'après les données d'usage, tu estimes qu'il touche environ **25 %** des nouveaux membres.

1. Combien d'entretiens faut-il pour avoir **90 %** de chances d'entendre ce problème au moins une fois ?
2. Combien pour l'entendre **au moins trois fois** avec 80 % de chances ? (Simule ou calcule avec la loi binomiale.)
3. Pour le sondage de confirmation : combien de répondants pour estimer la proportion à ± 5 points près (95 % de confiance, p = 25 %) ?

À toi, avec aide · réponse

Indices progressifs

Indice 1

Cherche le plus petit n tel que $1 - 0{,}75^n \geq 0{,}9$.

Indice 2

Pour 'au moins 3', calcule $1 - P(0) - P(1) - P(2)$ avec `math.comb`.

Indice 3

$n = (1{,}96 / 0{,}05)^2 \times p(1 - p)$.

À toi, avec aide · réponse

Réponse

```
import math
p = 0.25
n1 = next(n for n in range(1, 100) if 1 - (1 - p) ** n >= 0.90)
def au_moins(k, n): return 1 - sum(math.comb(n, i) * p**i * (1 - p)**(n - i) for i
    in range(k))
n2 = next(n for n in range(1, 100) if au_moins(3, n) >= 0.80)
n3 = math.ceil((1.96 / 0.05) ** 2 * p * (1 - p))
print(n1, n2, n3)
assert (n1, n2, n3) == (9, 16, 289)
```

1. **9 entretiens** suffisent pour entendre le problème au moins une fois avec 90 % de chances.
2. Pour l'entendre **au moins trois fois** avec 80 % de chances, il en faut **16** : la confiance en un thème (« plusieurs personnes l'ont dit ») coûte beaucoup plus d'entretiens qu'une simple détection.
3. **289 répondants** pour estimer la proportion à ± 5 points. Attention : une marge fine sur un échantillon mal choisi (les plus motivés répondent) reste trompeuse.

À toi, en solo

Analyseur de notes d'entretien. Écris un outil qui traite un corpus de 10 comptes rendus d'entretiens (que tu rédiges, 6 à 10 phrases chacun, avec des faits, des opinions, des promesses) et produit une synthèse :

1. classe chaque phrase (fait passé, engagement, opinion/promesse, solution proposée) avec la fonction du cours et calcule, par entretien, la **part de faits** (qualité de l'entretien) ;
2. regroupe les phrases en thèmes (mots-clés) et compte les **personnes** par thème (pas les citations) ;
3. identifie les **solutions proposées** (à ne pas retenir telles quelles) et remonte au problème sous-jacent par thème ;
4. marque les entretiens de faible qualité (moins de 40 % de faits) comme « à refaire » ;
5. calcule la saturation : combien de thèmes nouveaux apportent les entretiens 6 à 10 par rapport aux 5 premiers ;

À toi, en solo (suite)

6. produit un rapport texte : thèmes classés par nombre de personnes, avec 2 citations factuelles chacun.

À toi, en solo · réponse

Pistes de correction

```
import re
from collections import defaultdict

def classer(phrase):
    p = phrase.lower()
    if re.search(r"\b(je paie|j'ai payé|j'ai acheté|abonn)", p): return "engagemen
   t"
    if re.search(r"\b(j'ai|je suis allé|hier|la semaine dernière|le mois dernier|l
   a dernière fois|j'étais|je me suis|je me sentais|j'avais)\b", p): return "fait
   passé"
    if re.search(r"\b(je ferais|j'utiliserais|je voudrais|ça serait|ce serait|sûre
   ment|probablement)\b", p): return "opinion"
    if re.search(r"\b(il faudrait|pourquoi pas|vous devriez|et si)\b", p): return
   "solution proposée"
    return "autre"

THEMES = {"blocage": ("bloqué", "coincé", "perdu"), "temps": ("pas le temps", "fat
   igué", "le soir"), "motivation": ("abandonné", "lâché", "seul"), "prix": ("cher
   ", "abonnement")}
```

À toi, en solo · réponse

Pistes de correction (suite)

```
ENTRETIENS = {
 "E1": ["Je me suis retrouvé bloqué deux heures sur un exercice.", "J'ai demandé à
    un collègue, qui n'a pas répondu.", "Ça serait super d'avoir un forum."],
 "E2": ["La semaine dernière, j'ai abandonné un cours au bout de trois semaines.",
    "Je me sentais seul.", "J'utiliserais sûrement un groupe d'étude."],
 "E3": ["Je paie 30 € par mois pour une autre plateforme.", "Je n'ai pas le temps
   en semaine, seulement le soir.", "Il faudrait des cours plus courts."],
 "E4": ["Hier, j'ai été coincé sur une erreur incompréhensible.", "J'ai cherché su
   r un moteur de recherche pendant une heure."],
 "E5": ["Ça serait bien d'avoir un tableau de bord.", "Je ferais sûrement plus de
   cours avec des badges.", "Pourquoi pas un classement ?"],
 "E6": ["Le mois dernier, j'ai lâché un cours : j'étais fatigué le soir.", "J'ai c
   herché un résumé du chapitre, sans succès."],
 "E7": ["La dernière fois, je suis resté perdu dans le chapitre 3.", "J'ai posté s
   ur un forum externe, sans réponse.", "L'abonnement est cher."],
 "E8": ["Ce serait génial d'avoir un chat en direct.", "Probablement que je l'util
   iserais."],
 "E9": ["J'ai été bloqué trois fois ce mois-ci.", "Je paie un tuteur 40 € de l'heu
   re.", "J'étais seul face à l'erreur."],
```

À toi, en solo · réponse

Pistes de correction (suite)

```
 "E10": ["J'ai arrêté le cours de maths : trop long le soir.", "Je me suis senti s
   eul."],
}
qualite, theme_personnes, citations, vus = {}, defaultdict(set), defaultdict(list)
   , []
for e, phrases in ENTRETIENS.items():
    classes = [(p, classer(p)) for p in phrases]
    faits = [p for p, c in classes if c in ("fait passé", "engagement")]
    qualite[e] = len(faits) / len(phrases)
    themes_e = set()
    for p, c in classes:
        for t, mots in THEMES.items():
            if any(re.search(r"\b" + re.escape(m) + r"\b", p.lower()) for m in mot
   s):
                theme_personnes[t].add(e); themes_e.add(t)
                if c in ("fait passé", "engagement") and len(citations[t]) < 2: ci
   tations[t].append(f"{e} : {p}")
    vus.append(themes_e)
a_refaire = [e for e, q in qualite.items() if q < 0.40]
```

À toi, en solo · réponse

Pistes de correction (suite)

```
premiers = set().union(*vus[:5]); nouveaux = set().union(*vus[5:]) - premiers
print("à refaire :", a_refaire, "| thèmes nouveaux apportés par les entretiens 6-1
   0 :", nouveaux or "aucun")
for t, personnes in sorted(theme_personnes.items(), key=lambda kv: -len(kv[1])):
    print(f"{t:11s} {len(personnes)} personnes ← {citations[t]}")
assert {"E5", "E8"} <= set(a_refaire) and not ({"E4", "E6", "E7", "E9"} & set(a_re
   faire)) and theme_personnes["blocage"] >= {"E1", "E4", "E7", "E9"}
```

Les entretiens de faible qualité (E5, E8 : uniquement des opinions et des promesses) sont signalés « à refaire », avec les quelques entretiens où la part de faits reconnue par les règles simples est trop faible (le classeur par mots-clés est volontairement grossier : un humain relit). Les thèmes sont comptés en **personnes**, avec des citations **factuelles**. Les solutions proposées (forum, tableau de bord, classement) sont des **idées à valider**, pas des besoins : derrière « forum » se trouve le problème « je suis bloqué et personne ne répond ».

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Demander « utiliserais-tu ? »	Réponses polies, promesses sans valeur	Demander des faits passés et des engagements
N'interroger que des amis ou des fans	Échantillon biaisé	Recruter le segment visé, y compris les mécontents et les non-clients
Prendre une solution proposée pour un besoin	On construit la solution de quelqu'un d'autre	Remonter au problème (« pourquoi ? »)
Compter des citations, pas des personnes	Un bavard fausse le classement	Compter le nombre de personnes par thème

Mini-boss

Mini-boss

Synthèse de découverte de HashBox. Construis un jeu de données de 15 entretiens fictifs (10 à 12 phrases chacun, mélange de faits, opinions, promesses et solutions) sur le thème « abandon des nouveaux membres » et un sondage fictif de 400 réponses (graine fixée) sur 5 besoins (importance et satisfaction de 1 à 10).

1. évalue la qualité de chaque entretien (part de faits) et marque ceux à refaire ;
2. regroupe en thèmes, compte les personnes, et estime la saturation par courbe (thèmes nouveaux en fonction du nombre d'entretiens) ;
3. calcule l'opportunité de chacun des 5 besoins avec intervalle de confiance bootstrap sur les moyennes ;
4. construis un **arbre d'opportunités** (résultat visé → 3 opportunités → 2 idées chacune) sous forme de structure de données, et affiche-le ;
5. identifie l'opportunité à explorer en premier et rédige l'énoncé de problème (Qui / difficulté / cause / conséquence) ;

Mini-boss

Mini-boss (suite)

6. liste 3 biais possibles dans ton jeu de données et comment tu les atténuerais.

Mini-boss · réponse

Pistes de correction

Un bon dossier : chaîne de traitement entretien → qualité → thèmes → saturation; tableau d'opportunités avec intervalles qui se chevauchent ou non (la hiérarchie n'est affirmée que si les intervalles sont disjoints); arbre d'opportunités affiché en texte; énoncé de problème sans solution; biais cités (recrutement via la newsletter, biais de survie, politesse) avec contre-mesures (interroger des membres ayant abandonné, croiser avec les données d'usage).

Quiz éclair

Quiz éclair

1. Pour obtenir une information fiable en entretien, on demande plutôt :

A. si elle utiliserait notre idée
B. si elle trouve l'idée géniale
C. ce que la personne a fait la dernière fois

2. Le nombre de personnes à interroger se juge par :

A. la saturation : de nouveaux entretiens n'apportent plus de thème nouveau
B. un chiffre fixe de 100
C. le budget restant

3. L'opportunité d'un besoin est forte quand :

A. l'importance est faible

Quiz éclair

Quiz éclair (suite)

B. l'importance est forte et la satisfaction actuelle faible
C. la satisfaction est maximale

4. 🔁 *Révision mission 01* : Un résultat (outcome) est :

A. une fonctionnalité livrée
B. un fichier de spécifications
C. un changement observable dans le comportement des utilisateurs

5. 🔁 *Révision mission 01* : L'étoile du nord est :

A. une métrique qui reflète la valeur délivrée aux utilisateurs
B. le nombre de lignes de code
C. le chiffre d'affaires du trimestre précédent

Quiz éclair · réponse

Réponses

1. **C**.
2. **A**.
3. **B**.
4. **C**.
5. **A**.

Je réfléchis

1. Combien de fois, ce mois-ci, as-tu parlé à un utilisateur réel sans lui présenter ta solution ?
2. Quel besoin que tu crois « évident » n'a jamais été vérifié par un fait ?

À retenir

À retenir

1. Entretiens : le passé et les faits, pas les opinions ni les promesses.
2. Mom Test : parler de leur vie, pas de ton idée.
3. Thèmes par affinité, comptés en personnes ; saturation.
4. Sondage : marge d'erreur ≈ $1{,}96 \cdot \sqrt{p(1-p)/n}$, les biais restent.
5. Opportunité = importance + max(importance − satisfaction, 0) ; arbre d'opportunités.

MISSION ACCOMPLIE

+100 XP

Badge débloqué : Enquêteur de besoins

Prochaine mission : Hypothèses et validation

