COURS

Écrivez du code Python maintenable avec MVC et SOLID

P1C2 - Repérez ce qui rend un code difficile à maintenir

Vérifiez votre travail

1. La divergence. Les deux vues ne calculent pas le nombre de mois de la même façon. Pour Leïla, environ 219 jours se sont écoulés depuis son entrée, soit 7,3 mois de 30 jours. `balance` arrondit ce nombre au mois supérieur avec `math.ceil` : elle compte 8 mois, soit 16,64 jours de droits et un solde de 6,64 jours. `approve` le convertit avec `int()`, qui supprime la partie décimale : elle ne compte que 7 mois, soit 14,56 jours et un solde de 4,56 jours, insuffisant pour les 5 jours demandés. L'écart vaut exactement un mois de droits, 2,08 jours.

`int()` est un piège classique, justement parce qu'il ne ressemble pas à un arrondi : il sert d'ordinaire à convertir une valeur en entier. Pour un nombre positif, il revient pourtant à arrondir à l'unité inférieure. Une règle métier aussi importante mériterait d'être écrite explicitement, par exemple avec une division entière `// 30` accompagnée d'un nom qui dit ce qu'elle calcule.

Avez-vous aussi remarqué l'alternance ? Une copie écrit `25 + 5`, l'autre `30`. Le résultat est aujourd'hui identique, mais rien ne garantit que les deux écritures évolueront ensemble. C'est une divergence en germe.

2. Le calcul correct. La règle compte les mois complets : c'est l'arrondi au mois inférieur, donc `approve`, qui la respecte. Le bug vient de l'affichage, qui promet un solde que la salariée n'a pas. Leïla n'avait donc pas droit à ses 5 jours : le refus était juste, c'est l'affichage qui était faux.

3 et 4. Le diagnostic. Voici une proposition :

Problème	Signal	Conséquence pour l'ajout des stages	Type
`approve` vérifie, recherche, calcule, enregistre, notifie et répond	Responsabilités mélangées, fonction trop longue	La règle du stage doit être ajoutée dans une vue qui gère aussi la persistance et la notification	Organisation
Le calcul des droits est copié dans deux vues et les copies ont divergé	Duplication	La règle du stage doit être écrite deux fois, avec le risque d'une nouvelle divergence	Organisation
`NotificationService` et `date.today()` sont utilisés au cœur de la vue, et le calcul est mêlé aux accès à la base	Dépendances cachées	Impossible de tester la règle du stage sans base de données, sans notification réelle et pour une date choisie	Organisation

Vous avez pu retenir d'autres problèmes : la condition sur le type de contrat, la branche `else` qui attribue silencieusement zéro jour, les noms imprécis, les chaînes écrites en dur, le filtrage en Python après `objects.all()`, l'`IndexError` sur un identifiant inconnu, ou les erreurs renvoyées avec un statut 200, comme `{"error": "Employee not found"}`. Ils sont valables si vous les justifiez par un signal et une conséquence concrète. Les cinq derniers sont des problèmes locaux.

5. L'ordre de traitement. Commencez par rassembler le calcul des droits en un seul endroit : c'est ce qui supprime la cause du bug. Allégez ensuite les vues en déplaçant les règles métier dans un service, puis remplacez la condition sur le type de contrat. L'assemblage des dépendances et le filtrage en base viendront au chapitre 4.

La distinction attendue est la suivante : corriger l'arrondi traite le **symptôme** ; rassembler la règle en un seul endroit traite la **cause**. C'est cet argument qui permettra à Inès de justifier le temps consacré au refactoring.

Erreurs fréquentes

• **Se limiter à remplacer** `math.ceil` **par** `math.floor` : le bug disparaît, mais la règle reste dupliquée. La prochaine évolution, comme l'ajout des stages, pourra recréer le même écart.

• **Ne pas voir l'arrondi caché dans** `int()` : en cherchant un `math.floor` dans `approve`, vous concluez que la vue n'arrondit pas, alors que `int()` le fait déjà.

• **Choisir soi-même la bonne règle** : sans la confirmation de Samuel, vous auriez pu « corriger » l'approbation pour l'aligner sur l'affichage, et accorder des jours que les salariés n'ont pas.

• **Décrire un problème sans signal ni conséquence** : « ce code est mal écrit » ne permet pas à Inès de prendre une décision ; « la règle est dupliquée, donc le stage devra être ajouté deux fois » le permet.

• **Modifier le code pendant le diagnostic** : vous perdez votre point de comparaison, et les tests ne vous disent plus si le comportement a changé.