Refacto :

Ci dessous la liste des principales améliorations et optimisations apportées :

1. Utilisation de la déclaration **declare(strict\_types=1);** pour activer le typage strict.
2. Injection de dépendances : Les dépendances sont injectées via le constructeur plutôt que d'utiliser des singletons.
3. Typage fort : Ajout de types pour les paramètres et les valeurs de retour.
4. Séparation des responsabilités : La logique de remplacement des variables est découpée en méthodes plus petites et plus spécifiques.
5. Utilisation de **strtr()** pour les remplacements multiples, ce qui est plus efficace que plusieurs appels à **str\_replace()**.
6. Simplification de la logique conditionnelle en utilisant l'opérateur **null** **coalescing** (**??**).
7. Suppression des variables non utilisées et des conditions redondantes.
8. Utilisation de méthodes privées pour améliorer la lisibilité et la maintenabilité.
9. Remplacement de **and** par **&&** pour une meilleure lisibilité et cohérence avec les pratiques modernes.

Détails du refacto :

1. Injection de dépendances :

Au lieu d'utiliser des singletons **(getInstance())**, utilisation de l’ injection des dépendances via le constructeur. Ça améliore la testabilité et réduit le couplage.

public function \_\_construct(

QuoteRepository $quoteRepository,

SiteRepository $siteRepository,

DestinationRepository $destinationRepository,

ApplicationContext $applicationContext

) {

// ...

}

1. Séparation des responsabilités :

Découpage de la méthode computeText() en plusieurs méthodes plus petites et spécifiques pour rendre le code plus lisible et plus facile à maintenir

private function replaceQuoteVariables(string $text, Quote $quote): string

private function replaceUserVariables(string $text, User $user): string

1. Utilisation de strtr() pour les remplacements : plus efficace pour les remplacements multiples que plusieurs appels à str\_replace().

$replacements = [

'[quote:destination\_name]' => $destination->countryName,

// ...

];

return strtr($text, $replacements);

1. Simplification des conditions : Utilisation de l'opérateur null coalescing et de vérifications de type plus concises.

$quote = $data['quote'] ?? null;

if ($quote instanceof Quote) {

// ...

}

1. Typage fort : L'ajout de types pour les paramètres et les valeurs de retour améliore la robustesse du code et facilite sa compréhension.
2. Méthode privée pour la logique complexe : La logique de création du lien de destination a été extraite dans une méthode privée pour améliorer la lisibilité.
3. Suppression du code mort : Les variables non utilisées et les conditions redondantes ont été supprimées.
4. Cohérence des noms de variables : Les noms de variables sont standardisés pour plus de clarté ( $quoteFromRepository au lieu de $\_quoteFromRepository), principe de clean code.

* **Ces modifications rendent le code plus modulaire, plus facile à tester et à maintenir, tout en améliorant potentiellement les performances.**
* **Cette nouvelle structure facilite également l'ajout de nouvelles fonctionnalités ou la modification de la logique existante.**