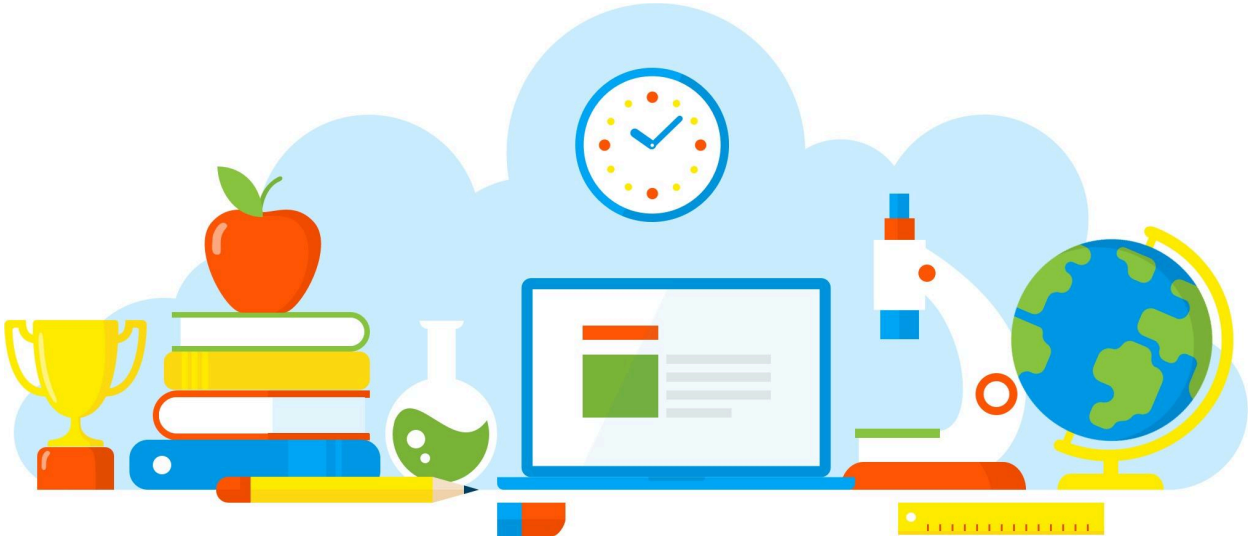

Plan établi pour réduire la dette technique de l'application

Todolist

Filâtre Killian
28/06/2024

Introduction

Ce document désigne l'accumulation de défauts ou de choix techniques qui ralentissent le développement et augmentent les coûts de maintenance de l'application Todolist. La gestion efficace de cette dette sera recommandée pour maintenir l'application performante et évolutive. Il décrit également le plan d'action mis en place pour réduire la dette technique de cette application.

Objectifs

1. Améliorer la performance et la stabilité de l'application.

2. Faciliter la maintenance et l'évolution du code.

3. Mettre à jour les technologies obsolètes.

4. Réduire les risques de sécurité.

Analyse

1. Analyse de l'État Actuel

1. **Audit du Code:**
    - Utilisation de l'outil symfony insight pour analyser la qualité du code.
    - Identification des points de dette technique comme les duplications de code, les méthodes dépréciées …
2. **Évaluation des Dépendances:**
    - Lister toutes les dépendances actuelles et vérifier les versions.
    - Identifier les dépendances obsolètes ou non sécurisées.

2. Mise à Jour des Technologies

1. **Mise à Jour de PHP:**
    - Passer de PHP 7.1 à une version supportée et stable (PHP 8.3).
    - Tester l'application sur la nouvelle version de PHP en utilisant un environnement de développement.
2. **Mise à Jour de Symfony:**
    - Planifier la migration de Symfony 3 à Symfony 6.4 (Dernière version stable).
    - Suivre les guides de migration officiels de Symfony.
    - Mettre à jour les bundles et composants de Symfony.

3. Refactorisation du Code

1. **Simplification et Modernisation:**
    - Réécrire les parties du code qui sont trop complexes ou non conformes aux standards actuels.
    - Utiliser les fonctionnalités modernes de PHP et de Symfony.
2. **Tests Unitaires et Fonctionnels:**
    - Augmenter la couverture des tests unitaires et fonctionnels.
    - Utiliser PHPUnit pour automatiser les tests.

4. Amélioration de la Sécurité

1. **Revue de Sécurité:**
    - Effectuer une analyse de sécurité complète.
    - Corriger les vulnérabilités identifiées (injections SQL, XSS …).
2. **Mises à Jour de Sécurité:**
    - Implémenter des pratiques de sécurité recommandées (validation des entrées, CSRF …).

5. Documentation et Formation

1. **Mise à Jour de la Documentation:**
    - Mettre à jour la documentation technique (Implémentation de l'authentification)

Planning

1. **Phase 1: Analyse (1 mois)**
    - Audit du code, évaluation des dépendances.

2. **Phase 2: Mise à Jour des Technologies (2 mois)**
    - Mise à jour de PHP et Symfony.

3. **Phase 3: Refactorisation du Code (3 mois)**
    - Simplification, augmentation de la couverture de tests (minimum 70%), optimisation des performances.

4. **Phase 4: Amélioration de la Sécurité (1 mois)**
    - Revue de sécurité, corrections...

5. **Phase 5: Documentation et Formation (1 mois)**
    - Mise à jour de la documentation.

Conclusion

La réduction de la dette technique est un processus continu qui nécessite d'investir du temps et des ressources. On vise à améliorer la qualité, la performance et la sécurité de l'application, tout en préparant le terrain pour des features.