Plan d'Implémentation de la Stratégie de Mise en Cache

Prérequis avant de commencer le développement

1. **Compréhension de l'architecture existante**
    - Revoir l'architecture des services actuels (CardStorageService, UserService, HistoryService, etc.)
    - Comprendre les flux de données et les appels Firebase existants
2. **Vérification des dépendances**
    - S'assurer que Angular 18+ et RxJS 7+ sont bien installés et configurés
    - Vérifier les versions compatibles des librairies Firebase utilisées
3. **Environnement de développement**
    - Mettre en place un environnement de développement avec outils de debug et monitoring
    - Configurer les outils de mesure de performance pour établir les métriques de base
4. **Documentation préalable**
    - Documenter les points d'appel Firebase actuels et leur fréquence
    - Établir les métriques de base (temps de chargement, nombre d'appels à Firebase)

Phases et Tâches d'Implémentation

Phase 1: Préparation et Architecture

Tâche 1.1: Analyse des services existants

- Identifier tous les services qui effectuent des appels à Firebase
- Analyser les patterns d'accès aux données (fréquence, volume, structure)
- Prioritiser les services selon leur impact potentiel (commencer par CardStorageService)

Tâche 1.2: Définition de l'architecture de cache

- Définir l'interface commune pour les méthodes de cache
- Créer des modèles pour les BehaviorSubjects et leur gestion
- Documenter la stratégie d'invalidation du cache

Tâche 1.3: Préparation des tests

- Mettre en place la structure des tests unitaires
- Définir les métriques de performance à suivre
- Créer des jeux de données de test représentatifs

Phase 2: Implémentation du Cache pour CardStorageService

Tâche 2.1: Refactoring initial du CardStorageService

- Ajouter les propriétés BehaviorSubject et Observable publique
- Implémenter le suivi de l'ID utilisateur pour le cache
- Ajouter la méthode de réinitialisation du cache

Tâche 2.2: Implémentation de la récupération avec cache

- Modifier getCardsByUserId pour vérifier d'abord le cache
- Implémenter la logique de chargement depuis Firebase si nécessaire
- Mettre à jour le BehaviorSubject avec les données récupérées

Tâche 2.3: Implémentation de l'ajout avec mise à jour du cache

- Modifier addCard pour mettre à jour Firebase puis le cache
- Implémenter la gestion d'erreur avec restauration de l'état précédent
- Assurer la création d'une nouvelle instance du tableau pour l'immutabilité

Tâche 2.4: Implémentation de la suppression/modification avec mise à jour du cache

- Modifier deleteCard et updateCard pour mettre à jour le cache après Firebase
- Implémenter la sauvegarde de l'état précédent pour le rollback en cas d'erreur
- Assurer la création de nouvelles instances pour l'immutabilité

Tâche 2.5: Intégration avec le service d'authentification

- Connecter clearCache au processus de déconnexion
- S'assurer que le cache est vidé lors du changement d'utilisateur
- Vérifier qu'aucune donnée de l'utilisateur précédent ne persiste

Phase 3: Extension à d'autres Services

Tâche 3.1: Mise en cache du UserService

- Appliquer le pattern BehaviorSubject au UserService
- Implémenter la récupération, mise à jour et réinitialisation du cache
- Intégrer la gestion des erreurs

Tâche 3.2: Mise en cache du HistoryService

- Appliquer le pattern BehaviorSubject au HistoryService
- Implémenter les méthodes de cache spécifiques à l'historique
- Intégrer la gestion des erreurs

Tâche 3.3: Mise en cache du CollectionHistoryService

- Appliquer le pattern BehaviorSubject pour l'historique de la collection
- Implémenter la récupération et mise à jour du cache
- Intégrer la gestion des erreurs

Phase 4: Optimisations et Gestion des Erreurs

Tâche 4.1: Amélioration de la gestion des erreurs

- Standardiser les messages d'erreur et les logs
- Implémenter les fallbacks pour chaque type d'erreur identifié
- Créer des composants UI pour les messages d'erreur et les options de rechargement

Tâche 4.2: Optimisation des performances

- Modifier les composants pour utiliser la stratégie OnPush de détection de changement
- Optimiser les abonnements aux Observables (unsubscribe approprié)
- Vérifier l'impact mémoire et ajuster si nécessaire

Tâche 4.3: Mise en place du monitoring

- Implémenter des métriques pour suivre le nombre d'appels à Firebase

- Mesurer les temps de chargement avec et sans cache
- Créer un dashboard de monitoring simple pour suivre les améliorations

Phase 5: Tests et Validation

Tâche 5.1: Tests unitaires

- Implémenter les tests unitaires pour chaque service avec cache
- Vérifier les scénarios de succès et d'erreur
- Tester la réinitialisation du cache et les changements d'utilisateur

Tâche 5.2: Tests d'intégration

- Tester l'interaction entre les services et les composants
- Valider que les mises à jour se propagent correctement
- Vérifier la cohérence des données entre les différentes vues

Tâche 5.3: Tests de performance

- Mesurer la réduction des appels à Firebase
- Comparer les temps de chargement avant/après l'implémentation
- Vérifier l'impact sur la mémoire et les performances de l'application

Cahier de Tests en format Gherkin

Feature: Cache des Cartes Pokémon

Scénario: Première consultation des cartes après connexion

```
Given un utilisateur vient de se connecter à l'application
When il accède à la page MyWallet pour la première fois
Then le service CardStorageService doit récupérer les cartes depuis Firebase
And stocker les cartes dans le BehaviorSubject
And afficher les cartes à l'utilisateur
```

Scénario: Navigation répétée vers la page des cartes

```
Given un utilisateur est connecté et a déjà consulté ses cartes
When il navigue vers une autre page puis revient à la page MyWallet
Then le service CardStorageService doit récupérer les cartes depuis le cache
And ne pas effectuer d'appel à Firebase
And afficher instantanément les cartes à l'utilisateur
```

Scénario: Ajout d'une nouvelle carte

```
Given un utilisateur est sur la page d'ajout de carte
When il remplit les détails d'une nouvelle carte et la soumet
Then le service CardStorageService doit sauvegarder la carte dans Firebase
And mettre à jour le BehaviorSubject avec la liste incluant la nouvelle carte
And toutes les vues abonnées doivent être automatiquement mises à jour
And l'utilisateur doit voir sa nouvelle carte sans rechargement complet
```

Scénario: Suppression d'une carte

```
Given un utilisateur est sur la page de détail d'une carte
When il choisit de supprimer la carte
Then le service CardStorageService doit supprimer la carte de Firebase
And mettre à jour le BehaviorSubject en retirant la carte du tableau
And toutes les vues abonnées doivent être automatiquement mises à jour
And l'utilisateur ne doit plus voir la carte sans rechargement complet
```

Scénario: Modification d'une carte

```
Given un utilisateur est sur la page de détail d'une carte
When il modifie les informations de la carte et sauvegarde
Then le service CardStorageService doit mettre à jour la carte dans Firebase
And mettre à jour le BehaviorSubject avec les nouvelles informations
And toutes les vues abonnées doivent être automatiquement mises à jour
And l'utilisateur doit voir les modifications sans rechargement complet
```

Scénario: Déconnexion et reconnexion avec un compte différent

```
Given l'utilisateur A est connecté et a consulté ses cartes
When il se déconnecte
Then le cache du CardStorageService doit être vidé
When l'utilisateur B se connecte et accède à la page MyWallet
Then le service doit charger les cartes de l'utilisateur B depuis Firebase
And aucune donnée de l'utilisateur A ne doit être visible
```

Feature: Gestion des Erreurs de Cache

Scénario: Erreur lors du chargement initial depuis Firebase

```
Given un utilisateur accède à ses cartes pour la première fois
When une erreur se produit lors de la récupération des données depuis Firebase
Then l'erreur doit être consignée dans les logs
And un message d'erreur approprié doit être affiché à l'utilisateur
And une option de rechargement manuel doit être proposée
When l'utilisateur choisit de recharger
Then une nouvelle tentative de récupération depuis Firebase doit être effectuée
```

Scénario: Erreur lors de l'ajout d'une carte

```
Given un utilisateur tente d'ajouter une nouvelle carte
When une erreur se produit lors de la sauvegarde dans Firebase
Then l'état précédent du cache doit être conservé
And un message d'erreur doit informer l'utilisateur du problème
And des options pour réessayer ou annuler doivent être proposées
```

```
When l'utilisateur choisit de réessayer
Then une nouvelle tentative d'ajout doit être effectuée
```

Feature: Performance du Cache

Scénario: Mesure de la réduction des appels Firebase

```
Given les métriques d'appels à Firebase ont été établies avant l'implémentation
When l'application est utilisée avec le système de cache pendant une session typique
Then le nombre total d'appels à Firebase doit être réduit d'au moins 70% par rapport aux métriques de base
```

Scénario: Mesure du temps de chargement

```
Given les métriques de temps de chargement ont été établies avant l'implémentation
When un utilisateur accède à ses cartes après la première consultation
Then le temps de chargement doit être inférieur à 100ms
And la réactivité doit être perçue comme instantanée
```

Scénario: Vérification de l'impact mémoire

```
Given l'application est en cours d'exécution avec le système de cache
When l'utilisation de la mémoire est mesurée pendant une session typique
Then l'augmentation de l'empreinte mémoire ne doit pas dépasser 5% par rapport à la version sans cache
```