

Atelier 4 - Conserver les résultats

Objectif : Charger, enregistrer, effacer et gérer les erreurs.

Travailler en binôme ; alterner pilote et observateur. Garder une copie de votre travail avant de consulter une correction.

1. Reprise du TP3 - Notre parcours (45 min au total)

- Montrer un parcours et une observation de test
- Expliquer navigate, replace ou popToTop
- Présenter le futur objet résultat

Reprise incluse : 30 min de revue du travail personnel, puis 15 min de transition.

Des questions ? Noter une correction et un test à rejouer.

2. Lire et écrire avec AsyncStorage (60 min, explications et questions comprises)

- Créer storage.js et une clé versionnée
- Lire l'historique au montage avec useEffect
- Réussite : vide au premier lancement, sans erreur

Point de contrôle : montrer le résultat et expliquer une ligne de code.

Des questions ? Noter ce qui bloque avant de poursuivre.

3. Protéger les transitions asynchrones (60 min, explications et questions comprises)

- Enregistrer un résultat puis relancer l'application
- Provoquer une erreur de lecture sur une copie
- Vérifier que rien n'est écrasé silencieusement

Point de contrôle : montrer le résultat et expliquer une ligne de code.

Des questions ? Noter ce qui bloque avant de poursuivre.

4. Supprimer et prouver la persistance (60 min, explications et questions comprises)

- Exécuter les cas stockage du cahier de recette
- Tester Annuler puis Confirmer la suppression
- Préparer une démonstration de 90 secondes

Point de contrôle : montrer le résultat et expliquer une ligne de code.

Des questions ? Noter ce qui bloque avant de poursuivre.

Aide progressive

1. Lire le message d'erreur et le fichier indiqué.
2. Comparer l'état attendu avec l'état observé.
3. Demander un indice au binôme voisin.
4. Demander à l'enseignant le point de reprise adapté.

À remettre

Votre code, une courte preuve de fonctionnement, les environnements testés et une difficulté expliquée. Le format et la date de remise seront annoncés en salle.

Manipulations guidées

1. Examiner `domain.mjs` : quel résultat est valide ? Écrire un exemple accepté et un exemple refusé.
2. Dans `storage.js`, utiliser une clé fixe versionnée, `getItem`, `setItem` et `removeItem`. Ne pas écrire un objet directement : le sérialiser.
3. Partir de la structure `HistoryProvider` fournie dans l'étape 05. Identifier les quatre états `history`, `ready`, `error` et `busy`. Les écrans consomment ce contexte partagé.
4. Lire dans `useEffect`. Après réussite seulement, passer `ready` à `true` ; en cas d'échec, afficher une erreur et conserver la possibilité de relire.
5. Sur `Résultat`, ajouter l'action `Enregistrer` et attendre la promesse avant de montrer le succès. Ne pas sauvegarder depuis le rendu du composant.
6. Afficher `history` avec `map` ; prévoir le message vide. Ajouter l'effacement confirmé et son annulation.
7. Suivre les injections d'erreur de `recette.pdf` sur une copie. Restaurer les méthodes avant de livrer.

Défi : afficher le meilleur score calculé depuis l'historique sans le stocker séparément.

Travail personnel et reprise

Lire `travail-personnel.pdf` (ressource 16) et copier `fiche-suivi-personnel.pdf` (17). Aucun devoir pendant le déjeuner ; reprise de dix minutes en S5.