Sommaire

1 Architecture Decision Record (ADR)
Relevé de Décision d'Architecture (RDA)
Geotrek-admin-mobile (GTAM)

Titre : Création d'une PWA dédiée pour la saisie terrain (Geotrek-admin-mobile)
Statut : Proposé
Date : 19 Février 2026

Les gestionnaires de sentiers (agents de parcs, conseils départementaux, etc.) ont un besoin récurrent d'effectuer des saisies sur le terrain (aménagements, signalétique, signalements...). Actuellement, bien que Geotrek-admin soit responsive, il ne permet pas de travailler hors-ligne.

Les fonctionnalités métiers prioritaires identifiées sont :

1. **Signalétique** (Création et modification)
2. **Interventions** (Modification puis création)
3. **Aménagements** (Création et modification)
4. **POI** (Création)

Le parc matériel cible est majoritairement constitué de smartphones et de quelques tablettes Android, avec une minorité d'appareils iOS.
La solution doit donc être multi-plateforme :

- La priorité est le fonctionnement sous Android.
- L'application peut être légèrement moins performante sous iOS, mais elle ne doit pas comporter de dysfonctionnements.

Il est nécessaire de pouvoir fonctionner sur des terminaux qui ne sont pas de toute dernière génération.
Certaines structures ont une gestion restrictive des terminaux dans le cadre de leur Mobile Device Management (MDM) : verrouillage de l'accès au google Play...

1.1 Décision

Nous décidons d'adopter le scénario consistant à développer une Progressive Web Application (PWA) dédiée spécifiquement à la saisie terrain.
Le code source de ce nouveau front end léger sera directement intégré et versionné au sein du dépôt GitHub de GeotrekCE.

La PWA exploitera les technologies web modernes (Service Workers, OPFS IndexedDB, Cache API) pour assurer un fonctionnement hors-ligne robuste.
Les décisions d'architecture tenteront de faciliter un encapsulage futur dans Capacitor ou PWABuilder.

Un socle API sécurisé en mode écriture sera développé côté serveur pour traiter les synchronisations.

1.2 Justification

Adéquation fonctionnelle : Les PWA sont bien gérées par Android (installation native via Chrome) et suffisamment bien gérées par iOS (via Safari) pour les cas, moins fréquents, d'usage sur iPhone, évitant ainsi de développer deux applications natives distinctes.
Les fonctionnalités nécessaires au bon fonctionnement de GTAM sont disponibles en PWA.

Coût et Maintenabilité : Cette approche évite de maintenir une nouvelle brique logicielle isolée. La PWA se mettra à jour de manière transparente pour les utilisateurs.

Complexité maîtrisée : Se concentrer sur une application front end allégée pour le terrain permet de limiter l'effort de développement.

C'est le plus simple de tous les scénarios étudiés.

Indépendance vis-à-vis d'acteurs externes : La distribution de la PWA se fait par simple URL, contournant les lourdeurs de publication, validation puis mises à jour récurrentes imposées par Google Play et l'App Store.

1.3 Conséquences et contraintes

Positives : Les agents pourront sélectionner une zone géographique hors-ligne pour ne synchroniser que le volume de données nécessaire. La prise de photos associée aux objets sera gérée hors-ligne.

API : Cela exige un effort de développement back end important pour transformer l'API existante (majoritairement en lecture) en une API transactionnelle capable de gérer les téléversements et les créations/modifications d'objets.

Les évolutions de l'API du back end de Geotrek admin doivent tenir compte non seulement du périmètre fonctionnel de GTAM mais aussi que toute l'interface pourrait être une PWA à terme.

Topologie et Segmentation dynamique : La complexité du référentiel linéaire empêche un calcul topologique en direct sur le smartphone hors-ligne.

Conflits de synchronisation : dans un premier temps, la gestion des conflits sera manuelle. En cas d'édition concurrente, la dernière modification enregistrée écrasera la version précédente. Dans tous les cas, c'est Geotrek admin qui gérera la synchronisation, pas l'application mobile qui se contentera d'envoyer seulement les champs modifiés afin de limiter les conflits.

Droits utilisateurs : La gestion des permissions s'appuiera sur geotrek-admin, qui définit déjà qui a le droit de voir / modifier / ajouter / supprimer des données par type

Il faudra déterminer si l'application utilise le mode clair/sombre en se calant sur les réglages du terminal, ou si l'utilisateur peut forcer le mode sombre à la demande.

2 Comparaison affinée des solutions natives et PWA

2.1 Contexte historique

Une première analyse comparant les différentes approches possibles avait conduit à quatre scénarios qui sont exposés dans l'Annexe 1 – « Historique de la première passe sur les différentes solutions possibles ».

Les inconvénients des solutions 1 (Qfield & co.) et 3 (tout Geotrek-Admin en PWA) sont suffisamment forts pour les écarter.

Il reste à comparer la réalisation en PWA ou application native.

Les choix non-techniques (coût, unicité des langages de développement, disponibilité des compétences, facilité de mise en œuvre…) poussent vers la PWA.

La PWA sera donc adoptée si elle a des performances techniques suffisantes.

On entend par « suffisantes » des performances permettant de réaliser les tâches assignées à GTAM. Il n'est donc pas nécessaire que la PWA soit en tout point aussi performante qu'une application native.

2.2 Ce qu'une PWA permet de faire mieux qu'une application native

La PWA est indépendant des évolutions des stores et des politiques d'Apple et Google :

- politique d'acceptation
- évolutions et mises à jour des SDK imposées
- contraintes contractuelles et financières
- …

De plus, GTAM étant une application destinée à un petit groupe d'utilisateurs et ne présentant pas d'intérêt pour le grand public, il est contre-productif de la publier sur des stores.

Le développement et la maintenance de la PWA permettent de réutiliser des compétences React similaires à celles mobilisées sur Geotrek Rando. Ce sont des compétences de développement web, plus communes que le développement mobile, ce qui augmente le nombre potentiel de (nouveaux) contributeurs éventuels.

La PWA peut être utilisée sur la plupart des terminaux disposant d'un navigateur : téléphones, tablettes, ordinateurs…

La mise à jour d'un PWA se fait en poussant simplement le nouveau site.
Côté terminal, la mise à jour de la PWA se fait en arrière-plan pour Android et à son ouverture pour iOS, il n'est pas nécessaire de publier des modifications vers les stores, de les faire accepter…

2.3 Possibilité de dépôt sur les stores

Il peut être intéressant d'anticiper la possibilité de déposer l'application sur les boutiques d'applications mobiles (Google Play et Apple App Store).
Pour ce faire, il est possible de réécrire l'application dans des langages natifs ou bien de l'encapsuler dans une surcouche dédiée.
Voir aussi :
Conversion de la PWA en app mobile · Issue #1091 · GeotrekCE/Geotrek-rando-v3
https://github.com/GeotrekCE/Geotrek-rando-v3/issues/1091

2.3.1 Encapsulage dans une surcouche

La méthode privilégiée actuellement est d'encapsuler la PWA dans une surcouche (*wrapper*) à l'aide de Capacitor ou de PWABuilder.
Pour cela, des contraintes doivent être respectées dans la conception de l'application, notamment :

- L'application doit être une SPA statique ; éviter le SSR.
- Les URL doivent être relatives.
- Créer une couche d'abstraction légère pour les API qui vérifie l'environnement dans lequel fonctionne l'application.
- Éventuellement utiliser les plugins Capacitor dès le départ (ex: @capacitor/geolocation). Ils fonctionnent aussi bien sur le Web que sur mobile. Cela évitera d'avoir à changer le code plus tard.
- Les appels réseaux doivent être traités en cross-origin.
- L'utilisation de service workers doit être limitée.
- Concevoir l'interface pour qu'elle ressemble à une application mobile plutôt qu'à un site web. Prévoir les sélecteurs CSS adéquats.
- Charger les ressources indispensables dans l'application plutôt que de les appeler par URL. Elles seront ainsi disponibles dès la première installation.
- Privilégier les jetons (JWT) plutôt que les cookies (souvent bloqués en cross-domain sur mobile).
- ...

Dans l'optique d'une application mobile, Capacitor ne gère pas OPFS pour le stockage de données. Il faut prévoir de pouvoir détecter le contexte et utiliser le système de fichiers dans le cas d'une application mobile.

2.3.2 Réécriture sous forme d'application native

L'amélioration rapide des outils s'appuyant sur de l'intelligence artificielle générative incite à penser qu'il est faisable de transformer rapidement une application existante en une autre application s'appuyant sur un autre langage.
Dans ce contexte, tranformer la PWA en application native semble techniquement réalisable. Cependant, nous ne disposons pas d'éléments permettant de dimensionner précisément le temps nécessaire. De plus, les changements radicaux que connaît actuellement ce secteur incitent à n'étudier la question qu'au moment de passer à la réalisation. Toute étude réalisée aujourd'hui serait probablement obsolète dans six mois, de même que les résultats d'une analyse réalisée il y a six mois ne sont plus valables aujourd'hui.

3 Adéquation des PWA

Différentes caractéristiques sont nécessaires au bon fonctionnement de GTAM.
Il n'était pas certain que les PWA offrent ces caractéristiques.
Voici une synthèse des différentes incertitudes qui ont été étudiées.

3.1 Une PWA peut-elle télécharger autant de données que nécessaire ?

La PWA doit stocker dans le téléphone des données cartographiques pour pouvoir les afficher en mode déconnecté.

Une PWA peut charger des fichiers pmtiles et GeoJSON de taille arbitraire en utilisant l'Origin Private File System (OPFS) :
https://developer.mozilla.org/en-US/docs/Web/API/File_System_API/Origin_private_file_system.
La limite de taille est constituée par l'espace disponible dans le terminal.
Le téléchargement de cartes de plusieurs Go a été testé sans problème sur
https://makinacorpus.github.io/maplibre-offline-pmtiles/#1.49/0/0.
Le chargement via OPFS est extrêmement rapide et s'approche des performances d'une application native, particulièrement en utilisant les SyncAccessHandles dans un Web Worker, qui permettent des lectures/écritures synchrones de bas niveau.

Pourquoi utiliser OPFS plutôt que IndexedDB or CacheStorage ?

- Performances très proches du natif
- Permet d'utiliser wasm et notamment SQLite wasm grâce aux performances supérieures et à l'accès synchrone
- OPFS est géré par le navigateur et invisible pour l'utilisateur. Il n'est donc pas nécessaire de lui demander l'autorisation de lecture ou écriture.

3.2 Peut-on informer l'utilisateur de l'espace dont il dispose pour enregistrer les données cartographiques ?

La PWA peut utiliser l'API navigator.storage.estimate(). Cette méthode renvoie l'espace total alloué à l'application (quota) et l'espace actuellement utilisé. Cela permet d'afficher une jauge d'espace libre à l'utilisateur avant le téléchargement de données cartographiques.
Ce n'est donc pas exactement tout l'espace disponible dans le terminal.

3.3 La persistance des données peut-elle être garantie ?

Risque d'effacement par le système : Par défaut, les navigateurs peuvent effacer les données stockées (y compris OPFS et IndexedDB) si le terminal manque d'espace de stockage.

Garantie de persistance : Pour éviter cela, la PWA doit demander le stockage persistant via navigator.storage.persist(). Si le navigateur (et l'utilisateur) accorde cette permission, les fichiers téléchargés (cartes) et les données saisies ne seront effacés que si l'utilisateur vide manuellement le cache de son navigateur ou utilise une fonction de suppression au sein de l'application.

3.4 Est-il possible d'effacer les données embarquées ?

La PWA peut fournir une interface aux API de stockage (OPFS, IndexedDB) permettant à l'utilisateur de supprimer des zones cartographiques spécifiques ou des fichiers pour libérer de l'espace.
Cela peut être utile pour charger de nouvelles données.

3.5 Performances d'affichage

Les interactions avec MapLibre GL JS dans une PWA sont presque aussi fluides qu'avec MapLibre dans une application native.

Tests réalisés : Affichage de cartes de la taille d'une région et superposition de données en GeoJSON.
L'affichage avec MapLibre GL JS est légèrement plus lent qu'avec une application native.
L'utilisateur ne perçoit la différence qu'en testant avec des terminaux côte à côte.
L'affichage dans la PWA reste rapide et est parfaitement utilisable.

Prévoir de confier le décodage PMTiles à un Web Worker pour ne pas bloquer le thread principal de l'UI.

3.6 La consommation est-elle raisonnable ?

La consommation d'une PWA est légèrement supérieure à celle d'une application native.
Huber, Stefan, Lukas Demetz, and Michael Felderer. "A comparative study on the energy consumption of Progressive Web Apps." Information Systems 108 (2022): 102017. https://doi.org/10.1016/j.is.2022.102017

L'utilisation prolongée du GPS et le rendu WebGL continuent d'être légèrement plus énergivores en PWA qu'en natif (les couches d'abstraction du navigateur consomment un peu plus de CPU/RAM).

Il faudra mettre en œuvre plusieurs stratégies d'optimisation :

- L'écran est généralement la première source de consommation :
  - Réduire la consommation de l'écran en maximisant les contrastes pour permettre une utilisation en plein air sans pousser au maximum la luminosité de l'écran.
  - Prévoir un mode sombre qui peut significativement faire diminuer la consommation pour les écrans OLED/AMOLED par forte luminosité.
    - Dash, Pranab, and Y. Charlie Hu. "How much battery does dark mode save? an accurate oled display power profiler for modern smartphones." *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 2021. https://doi.org/10.1145/3458864.3467682
    - Flores-Martin, Daniel, Sergio Laso, and Juan Luis Herrera. "Enhancing smartphone battery life: A deep learning model based on user-specific application and network behavior." *Electronics* 13.24 (2024): 4897. https://doi.org/10.1016/j.pmcj.2021.101384
    - Nugroho, Kuntoro Adi, and Shanq-Jang Ruan. "Saliency and power aware contrast enhancement for low OLED power consumption." *IEEE Transactions on Instrumentation and Measurement* 73 (2024): 1-17. https://doi.org/10.1109/TIM.2024.3350145
- La puce GPS consomme également beaucoup lors de son utilisation. Il faudra explorer des pistes pour la solliciter seulement en cas d'absolue nécessité
  - Ne pas l'utiliser : enableHighAccuracy: false
  - Réduire la fréquence de rafraîchissement du GPS, voire ne la lancer qu'à la demande. Utiliser getCurrentPosition (à la demande) plutôt que watchPosition (permanent).
  - Ne plus mettre à jour la position quand l'application est en arrière-plan.
  - Utiliser le capteur de mouvement pour relancer un positionnement
- Agir sur les autres services par exemple les service workers : Suspendre ceux qui peuvent l'être lorsque l'application passe en arrière-plan.

Avant d'investir trop de temps dans l'optimisation poussée de la consommation, il pourrait être intéressant de déterminer si l'autonomie pose un problème en conditions réelles d'utilisation.
On pourra donc revenir sur le sujet si la consommation est un obstacle majeur d'après les retours d'expérience au bout de plusieurs mois d'utilisation.

3.7 Le GPS en mode déconnecté fonctionne-t-il correctement dans une PWA ?

Précision : La précision du GPS dans une PWA (via l'API Geolocation) est identique à celle d'une application native, car le navigateur interroge les mêmes services de localisation du système d'exploitation.

Mode déconnecté : La puce GPS matérielle fonctionne indépendamment du réseau. En mode déconnecté, la PWA recevra toujours les coordonnées exactes. Seul le Time-To-First-Fix (temps pour obtenir la première position) peut être légèrement plus long sans le réseau (absence d'A-GPS), que ce soit pour une PWA ou une application native.

3.8 Comment les données saisies dans la PWA peuvent-elles être stockées dans le terminal ?

Les données saisies dans les champs de la PWA pourront être enregistrées dans IndexedDB.
Les photos prises avec l'appareil photo du terminal (via l'API MediaDevices ou un <input type="file">) seront gérées par OPFS.

3.9 Le téléversement de données fonctionne-t-il ?

Les données saisies dans une PWA peuvent être téléversées (upload) vers une API REST.
L'application offrira un bouton de synchronisation.

Pour ne rien lancer sans l'aval conscient de l'utilisateur, il n'y aura pas de synchronisation automatique (par exemple téléversement automatique lorsque la connectivité est rétablie).
De toute façon, la synchronisation automatique n'est pas gérée de façon uniforme. Mieux vaut ne pas l'inclure dans l'application :

- Sous Android+ Chrome, le téléversement des données saisies peut être lancé automatiquement dès qu'un accès Internet est disponible, en utilisant l'API *Background Sync*. Cette fonctionnalité n'est pas disponible sous Firefox
- Sous iOS, il n'existe pas d'équivalent. Il faudra attendre que l'utilisateur ouvre l'application pour déclencher la synchronisation.

Des mécanismes peuvent être mis en œuvre pour gérer les erreurs de téléversement :

- Conserver les données en local tant que le serveur n'a pas répondu par un statut HTTP de succès (200/201).
- Créer un mécanisme d'essais multiples avec un délai exponentiel.
- Signaler à l'utilisateur un échec définitif.

3.10 Est-il possible d'enregistrer dans l'écran d'accueil du téléphone des raccourcis vers des actions de l'application ?

L'application dispose d'une icône sur les écrans d'accueil. Certains *launchers* non d'origine ne gèrent pas ces icônes.
Il pourrait être intéressant de disposer de raccourcis vers des actions spécifiques (eg « Téléverser les saisies »).
Contrairement aux applications natives, il n'est pas possible de placer des icônes vers des actions spécifiques de l'application.
Sous Android, on peut cependant afficher un menu contextuel en faisant un appui long sur l'icône de la PWA (ex : « Nouvelle saisie », « Voir la carte », « Prendre une photo »...).
Sous iOS, ce n'est pas toujours possible en fonction de la version.

Les raccourcis (shortcuts) sont définis dans le fichier manifeste.
Une fois la PWA installée, la modification des raccourcis s'opère lors de la mise à jour de l'application.

3.11 Est-il possible de lancer la PWA en cliquant sur une URL ?

Chaque vue de l'application a une URL.
Si la PWA est installée, cliquer sur un lien reçu par SMS ou e-mail peut ouvrir directement l'application PWA plutôt qu'un onglet de navigateur classique.
Il faut pour cela configurer la PWA pour gérer la capture de liens et s'assurer que le navigateur l'autorise.

3.12 Choix du langage de développement

React qui est utilisé pour Geotrek Rando dispose d'un écosystème très riches pour les PWA.
D'autres frameworks (Svelte, SolidJS...) pourraient apporter des améliorations dans un domaine ou un autre (créer des bundles plus petits, être plus rapide pour le lancement initial...), mais aucun ne dispose de l'écosystème de React.

3.13 Des notifications sont-elles disponibles en PWA ?

S'il y a besoin de notifications :

- Android le permet (Web Push API)
- Sur iOS (à partir de la version 16.4), les notifications ne fonctionnent que si la PWA est ajoutée à l'écran d'accueil. Le processus pour demander la permission nécessite une interaction explicite de l'utilisateur.

Exemples d'utilisation de notifications :

- Informer l'utilisateur qu'une synchronisation massive en arrière-plan est terminée.
- Alerter d'une tâche à effectuer (eg téléverser les saisies).

3.14 Des versions minimales des OS et navigateurs sont-elles nécessaires ?

Des technologies clés comme OPFS (apparu vers 2022/2023) ne fonctionneront pas sur des versions d'OS trop anciennes.
Des terminaux plus anciens que 2022 peuvent bien sûr avoir été mis à jour avec des versions d'iOS récentes.

3.15 Inconvénients incontournables (iOS)

Bien qu'Apple ait amélioré le support PWA, des faiblesses demeurent par rapport à Android :

- Installation manuelle : Safari ne propose pas de bannière d'installation native invitant l'utilisateur à installer l'application. L'utilisateur doit manuellement ouvrir le menu de partage et sélectionner « Sur l'écran d'accueil ».
- Pas de synchronisation en arrière-plan : absence de la *Background Sync API*, obligeant l'application à être au premier plan pour synchroniser les données offline vers l'API.
- Éviction du stockage plus agressive : le moteur WebKit d'Apple est historiquement plus prompt à supprimer les données du cache et d'IndexedDB sans prévenir si l'utilisateur ne se sert pas de l'application pendant quelques semaines.

3.16 Installation de l'application

Le concept de PWA et d'installation n'est pas bien compris par le grand public.
GTAM s'adresse à un groupe réduit d'utilisateurs professionnels à qui une information adéquate devra être fournie.

On fournira également des informations pertinentes dans l'interface de l'application.
Voir la discussion autour des problèmes rencontrés pour Geotrek Rando :
https://github.com/GeotrekCE/Geotrek-rando-v3/issues/550

4 Annexe 1 – Historique de la première passe sur les différentes solutions possibles

L'objectif de cette revue est d'analyser les différentes solutions technologiques permettant de doter Geotrek-admin d'un outil mobile fonctionnant hors-ligne pour la collecte de données sur le terrain.

4.1 Scénario 1 : Utiliser un outil générique existant comme Qfield, Mergin Maps ou ODK

Description :
Utilisation d'un outil pré-existant dédié à la saisie de données en mobilié
La première piste suggère de s'appuyer sur des outils génériques de collecte de données mobiles existants, tels que Qfield, Mergin Maps (liés à QGIS) ou OpenDataKit (ODK), en les connectant à l'API de Geotrek-Admin ou directement à sa base de données. Cette approche présente l'avantage apparent de capitaliser sur des solutions éprouvées et riches en fonctionnalités de collecte de données géoréférencées.

Cependant, l'intégration avec QField, par exemple, soulève d'importantes complexités. Elle implique un lien étroit et continu avec l'écosystème QGIS, tant pour la préparation des projets de collecte (configuration des formulaires, symbologie, fonds de carte pour le mode hors-ligne) que pour la synchronisation des données. Le paramétrage des projets QField pour refléter la richesse et la complexité du modèle de données de Geotrek-Admin (avec ses nombreux modules, ses relations, sa logique métier et sa segmentation dynamique) est une tâche ardue et chronophage, voire impossible pour certains modules. Notre expérience, notamment sur des projets comme GeoRivière qui ont exploré cette piste pour la saisie terrain, nous a démontré que maintenir une synchronisation bidirectionnelle robuste et intuitive entre une base de données centrale complexe et des projets QField à grande échelle s'avère difficilement viable et engendre une charge de support et de formation considérable pour les utilisateurs finaux. Au final, le projet ne provoque pas d'adhésion et ce même en ce tournant vers des solutions telles que Qfield Cloud. Ces derniers doivent souvent acquérir une double compétence, sur Geotrek-Admin et sur QGIS/QField. De plus, la logique métier spécifique à Geotrek, notamment les contraintes liées à la segmentation dynamique lors de la création ou de la modification d'objets linéaires, serait particulièrement difficile à transposer et à garantir dans un outil générique.
Des outils comme ODK, bien que flexibles, nécessiteraient également un effort de configuration substantiel et pourraient ne pas offrir le même niveau d'intégration cartographique intuitive attendu par les utilisateurs de Geotrek.
La maintenabilité de la solution en fonction de la version de Geotrek représente également un défi important.

Ainsi, bien que ces outils soient puissants, notre expérience sur le sujet nous laisse penser qu'ils ne sont pas adaptés à l'envergure d'un projet tel que Geotrek et à ses contraintes propres (généricité des modèles, multi-structure, segmentation dynamique, etc.).

Avantages :
- Outils éprouvés pour la collecte de données géo-référencées.
- Outils maintenus par des communautés.

Inconvénients :
- La logique métier de Geotrek, notamment la segmentation dynamique, est difficilement transposable.
- L'expérience utilisateur est souvent dégradée, car elle demande de s'adapter à deux types d'interface et de modes de fonctionnement.
- La configuration des formulaires et la synchronisation des données peut être complexe.

Coûts :

- Coûts cachés importants en configuration des projets de collecte puis assistance et formation.

4.2 Scénario 2 : Développer une application mobile dédiée (native Android ou hybride)

Description : Développement d'une application native (Android/iOS) ou hybride (React Native, Flutter).

Avantages :
Expérience utilisateur parfaitement adaptée au mobile et accès optimisé aux fonctionnalités matérielles (GPS, appareil photo).
Facilité de stockage des données cartographiques et des données saisie
Fluidité de l'affichage cartographique (performances optimales)
Mode déconnecté plus efficace qu'avec une PWA
Notification

Inconvénients

- Implique la création et l'entretien d'une nouvelle brique logicielle distincte.
- Demande la maîtrise de langages de développement spécifiques, différents de ceux utilisés dans le reste des applications de Geotrek
- Le processus de publication sur les stores ajoute des contraintes de déploiement et de mises à jour régulières, avec une fréquence supérieure à celle des applications web/PWA
- Impose la création d'une API robuste en mode écriture (POST, PUT, DELETE).
- Le calcul des impacts topologiques de la segmentation dynamique ne peut pas se faire hors-ligne et s'effectue à la re-synchronisation.

Coûts :

- Coût initial de développement semble plus élevé que les solutions PWA particulièrement en raison de la montée en compétence et du temps de dépôt sur les stores. Le développement pur de l'application devrait être plus rapide qu'avec une PWA.
- Coûts de maintenance à long terme plus élevés.

4.3 Scénario 3 : Faire évoluer Geotrek-Admin sous forme de PWA

Description : Transformer l'intégralité de l'application web Geotrek-admin actuelle en Progressive Web App (PWA).

Cette approche repose sur un découplage total entre le backend et le frontend communiquant exclusivement par une API.

Avantages :

- Évite la multiplication des briques logicielles et offre une expérience utilisateur cohérente entre le bureau et le terrain.

Inconvénients :

- Implique une refonte majeure de Geotrek-admin vers une architecture API-centric. Cette refonte est souhaitable mais coûteuses et complexe.
- L'application résultante risque d'être trop lourde et confuse en raison de la présence de toutes les fonctionnalités de Geotrek-admin, dont beaucoup n'ont pas d'intérêt sur le terrain.

Coûts :

- L'ampleur de cette refonte rend cette option très coûteuse et complexe à court et moyen terme.

4.4 Scénario 4 : Une PWA dédiée à la saisie terrain pour ne pas tout refaire

Description :
Développer une nouvelle application front end légère(PWA) spécifiquement dédiée à la consultation et à la saisie sur le terrain.

Avantages :

- Les mises à jour de Geotrek-admin incluent automatiquement la dernière version de cette PWA.
- Elle est installable sur tout type de terminal sans passer par les stores d'applications.
- Elle fait appel aux mêmes technologies et à des savoir-faire proches de ceux de Geotrek Rando (JavaScript, React).

Inconvénients :

- Impose la création d'une API robuste en mode écriture (POST, PUT, DELETE).
- Le calcul des impacts topologiques de la segmentation dynamique ne peut pas se faire hors-ligne et s'effectue à la re-synchronisation.

Coûts
:

- Coûts optimisés par rapport à une refonte complète ou une application native.
- Les frais de maintenances sont les plus faibles