Université de Montréal

IFT3150 — Rapport final

Pique-Me : Montréal à la carte

Par

Tunwend-raabo Fahîma Carmen DABO (20266362)
Abdelghafour Rahmouni (20246224)
Naromba Condé (20251772)

Travail présenté à **Gena Han**
Dans le cadre du cours IFT3150 : Projet d'informatique

7 août 2025

Table des matières

1 Introduction

Contexte

À Montréal, les parcs font partie intégrante du quotidien : lieux de détente, de sport, de rencontres familiales ou d'événements de quartier. Lors des beaux jours, ces espaces deviennent des pôles d'activité majeurs. Pourtant, il n'existe toujours pas de moyen numérique unique et fiable pour connaître les différents espaces, se renseigner sur l'accessibilité ou la disponibilité d'un équipement avant de s'y rendre.

Problématique

Les usagers se posent invariablement les mêmes questions : *Le parc autorise-t-il les barbecues ? Est-il accessible aux personnes à mobilité réduite ? Reste-t-il des emplacements libres ? Comment trouver le parc qui repond le mieux à nos préférences ?* Les informations sont dispersées ou incomplètes ce qui crée une planification laborieuse et une expérience parfois décevante.

Proposition

Pique-Me vise à concentrer toutes ces données dans une application mobile intuitive : carte interactive, fiches synthétiques de parcs, filtres personnalisés, réservation d'emplacements ou d'activités, système d'avis/badges et notifications d'événements.

Objectifs

— Concevoir une **interface mobile fluide** listant les parcs, leurs équipements et les activités qui y sont proposées.
— **Implémenter la réservation** d'emplacements/activités avec double confirmation et annulation automatique.
— Développer une **dimension communautaire** (favoris, avis, badges).
— Satisfaire des exigences de **sécurité, performance, accessibilité** et disponibilité.

2 Analyse des besoins et étude préliminaire

2.1 Démarche générale

Le recueil des besoins a débuté dans le cadre du cours **IFT2905 – Interfaces personne-machine**, où *Pique-Me* a été choisi comme projet de session. Nous avons ensuite :

— réalisé des entretiens informels avec des amis et proches utilisateurs réguliers des parcs montréalais ;

— analysé nos propres expériences de planification de pique-niques et d'activités extérieures (difficultés et attentes).

Les résultats ont été triés puis convertis en exigences fonctionnelles et non fonctionnelles.

2.2 Exigences fonctionnelles

F1. Authentification & profil

- Inscription par courriel/Google ou usage invité ;
- Initialisation des préférences (activités, équipements) ;
- Consultation, modification, suppression du compte.

F2. Recherche & découverte de parcs

- Barre de recherche (adresse/nom) avec complétion automatique ;
- Filtres dynamiques (équipements, BBQ, activités proposées, etc.) ;
- Carte interactive dynamique : la carte suit la position de l'utilisateur et affiche progressivement les parcs (pins) visibles selon le niveau de zoom.

F3. Réservation

- Sélection d'emplacement ou d'activité via un calendrier ;
- Double confirmation (avant/après début) ;
- Annulation automatique si non-confirmation 60 min après le debut de la réservation.

F4. Interaction communautaire

Avis post-visite : L'utilisateur sélectionne un ou plusieurs badges pour recommander le parc selon son expérience (ex. plein air, famille, détente). Il peut aussi ajouter un commentaire et joindre des photos. Si un badge atteint un certain seuil de votes, une médaille (bronze, argent ou or) est attribuée au parc pour ce badge.

F5. Favoris et notifications

L'utilisateur peut ajouter ou retirer un parc de sa liste de favoris, qu'il peut consulter à tout moment pour retrouver rapidement ses lieux préférés. Des notifications personnalisées sont envoyées en fonction de ses activités : rappels de réservation, invitations

à répondre à un sondage post-visite, ou encore annonces d'événements organisés dans les parcs qu'il a mis en favori.

2.3 Exigences non fonctionnelles

Performance L'application doit être rapide : la plupart des requêtes doivent s'afficher en un temps minime. L'écran d'accueil doit être mis à jour automatiquement en fonction des préférences et de la localisation de l'utilisateur.

Sécurité Les connexions doivent être sécurisées, et les mots de passe bien protégés. Seules les personnes autorisées peuvent accéder aux données.

Compatibilité L'application doit bien fonctionner sur les téléphones Android et iPhone, avec des écrans de tailles standards.

Accessibilité L'interface doit être facile à utiliser, même pour les personnes ayant des difficultés visuelles ou motrices (bons contrastes, navigation simple, etc.).

Fiabilité L'application doit fonctionner sans planter, et rester disponible au moins 99 % du temps.

Confidentialité Seules les informations nécessaires sont demandées. L'utilisateur peut à tout moment effacer son compte ou ses données s'il le souhaite.

2.4 Méthodologie retenue

Pour développer notre application *Pique-Me*, nous avons choisi une approche Agile en utilisant la méthode Scrum. Cette méthode permet d'avancer de manière itérative, en découpant le projet en sprints de une semaines. Chaque sprint se concentre sur un ensemble clair de fonctionnalités à livrer, facilitant la planification, l'adaptation continue et l'évaluation du progrès.

Cette approche nous permet de rester flexibles, réactifs aux retours, et de garantir un produit fonctionnel à chaque étape importante du projet.

3 Conception

3.1 Architecture du système

L'application *Pique-Me* repose sur une architecture en trois couches, chaque composant ayant un rôle spécifique dans le traitement des données et l'interaction utilisateur.

— **Frontend :** développé avec React Native et Expo, il s'agit de l'interface utilisateur de l'application mobile (Android/iOS). Elle permet de :

  — rechercher des parcs et filtrer les résultats,

  - réserver des emplacements ou activités,
  - consulter ses favoris et gérer son compte.
- **Backend :** basé sur Node.js et Express, il traite les requêtes envoyées par l'application. Il centralise la logique métier, vérifie les disponibilités, contrôle les réservations, interroge Firebase et l'API de la Ville de Montréal.
- **Données :** deux sources principales :
  - **Firebase :** stocke les utilisateurs, réservations, favoris, avis, etc.
  - **API de la Ville de Montréal :** fournit les données officielles des parcs (équipements, localisation, règlements). Ces données ont été extraites et stockées dans Firebase.

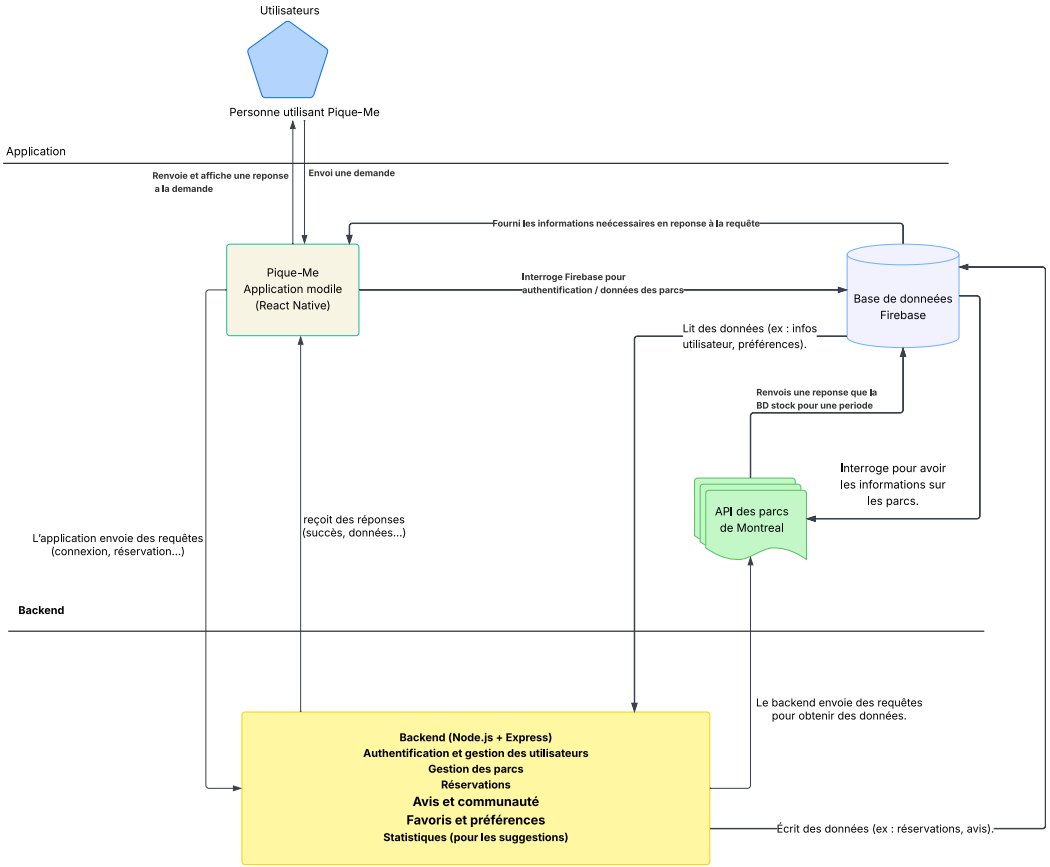

FIGURE 1 – Architecture logique de l'application Pique-Me.

3.2 Modèle de données

Le schéma suivant montre le modèle de données UML utilisé pour l'application Pique-Me. Il présente les éléments importants de l'application, comme les utilisateurs, les parcs, les réservations, les événements, les équipements, les avis et les photos. On y voit aussi comment ces éléments sont liés entre eux.

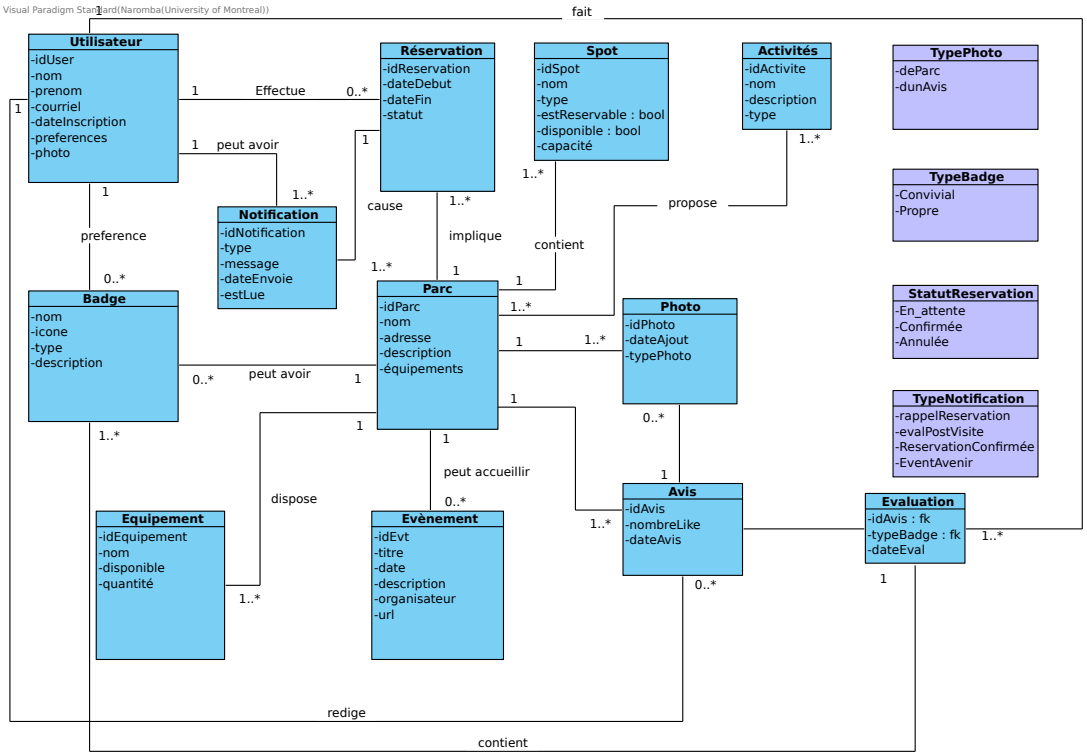

FIGURE 2 – Diagramme Entité–Association de Pique-Me.

3.3 Choix technologiques et justifications

— **React Native + Expo :** développement multiplateforme rapide, composants UI réutilisables, configuration simplifiée.

— **Node.js + Express :** création d'une API REST structurée et découplée de l'interface.

— **Firebase :** base de données temps réel, authentification sécurisée, simplicité de déploiement.

— **API Ville de Montréal :** source officielle des données des parcs.

— **Expo Location :** géolocalisation pour rechercher les parcs à proximité.

3.4 Diagrammes d'activités (flux utilisateurs)

3.4.1 Inscription et connexion

Ce diagramme illustre les étapes principales du flux d'inscription et de connexion de l'utilisateur.

3.4.2 Recherche et filtrage de parcs

Ce diagramme montre le processus de recherche et de filtrage des parcs par l'utilisateur.

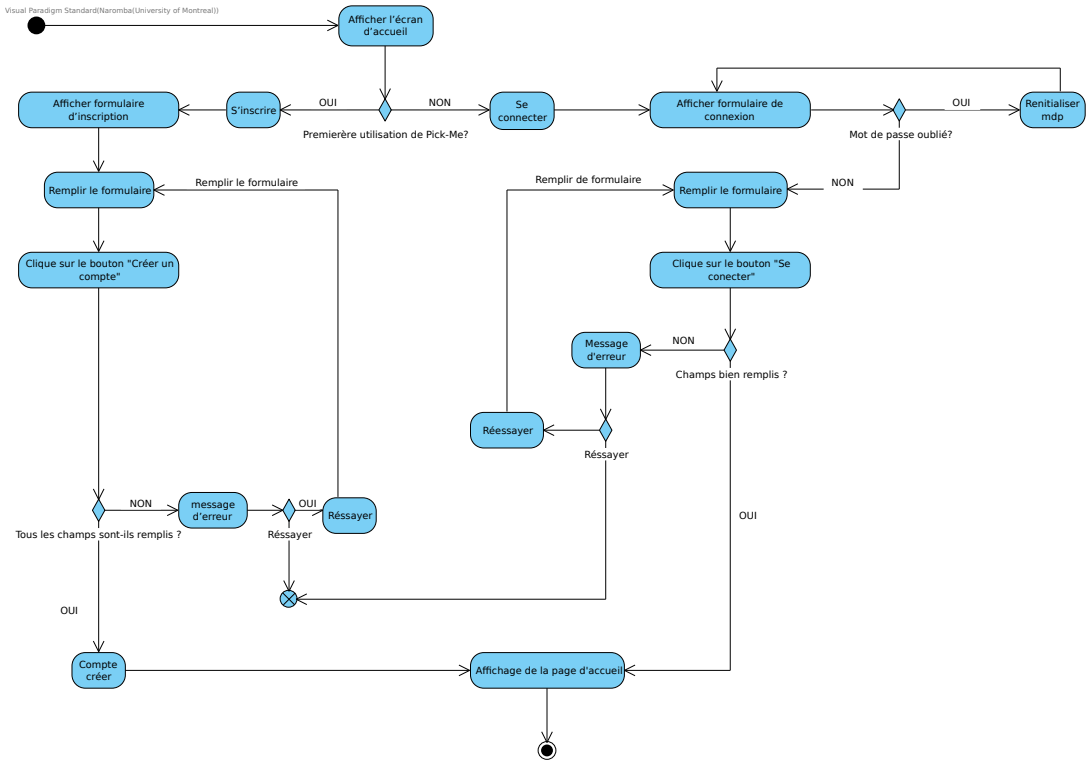

FIGURE 3 – Flux : Inscription et connexion

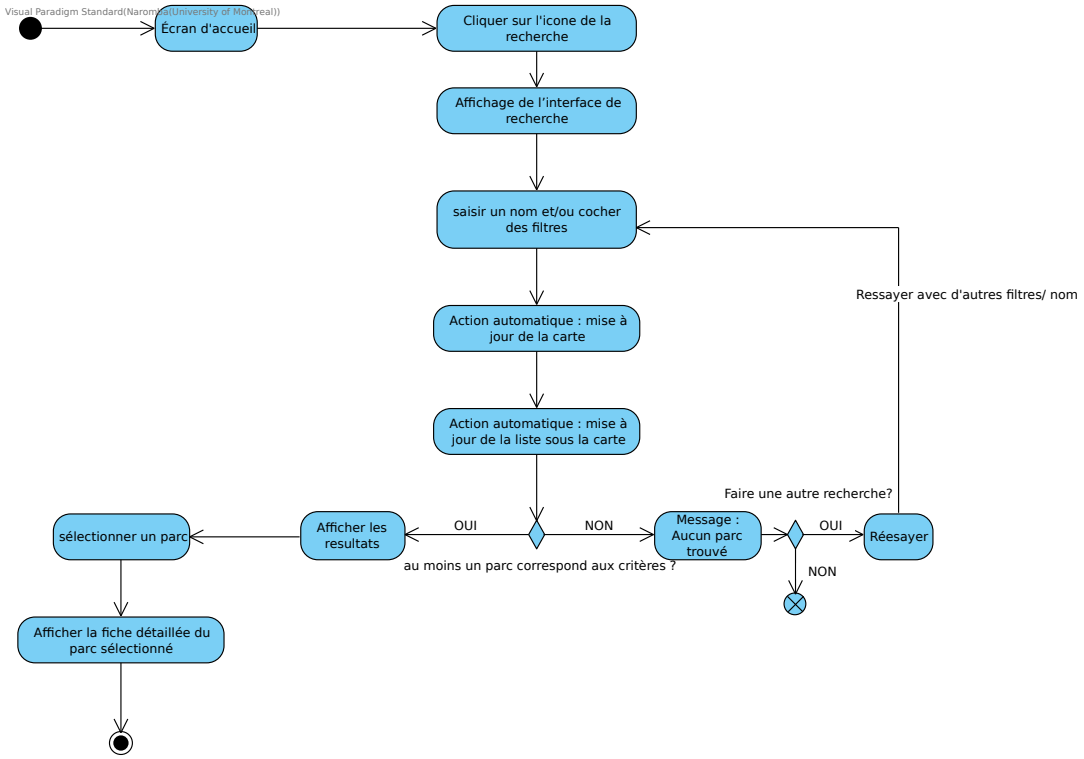

FIGURE 4 – Flux : Recherche et filtrage de parcs

3.4.3 Réservation d'un emplacement

Parcours utilisateur pour choisir un emplacement sur la carte du parc, sélectionner une date/heure puis confirmer la réservation.

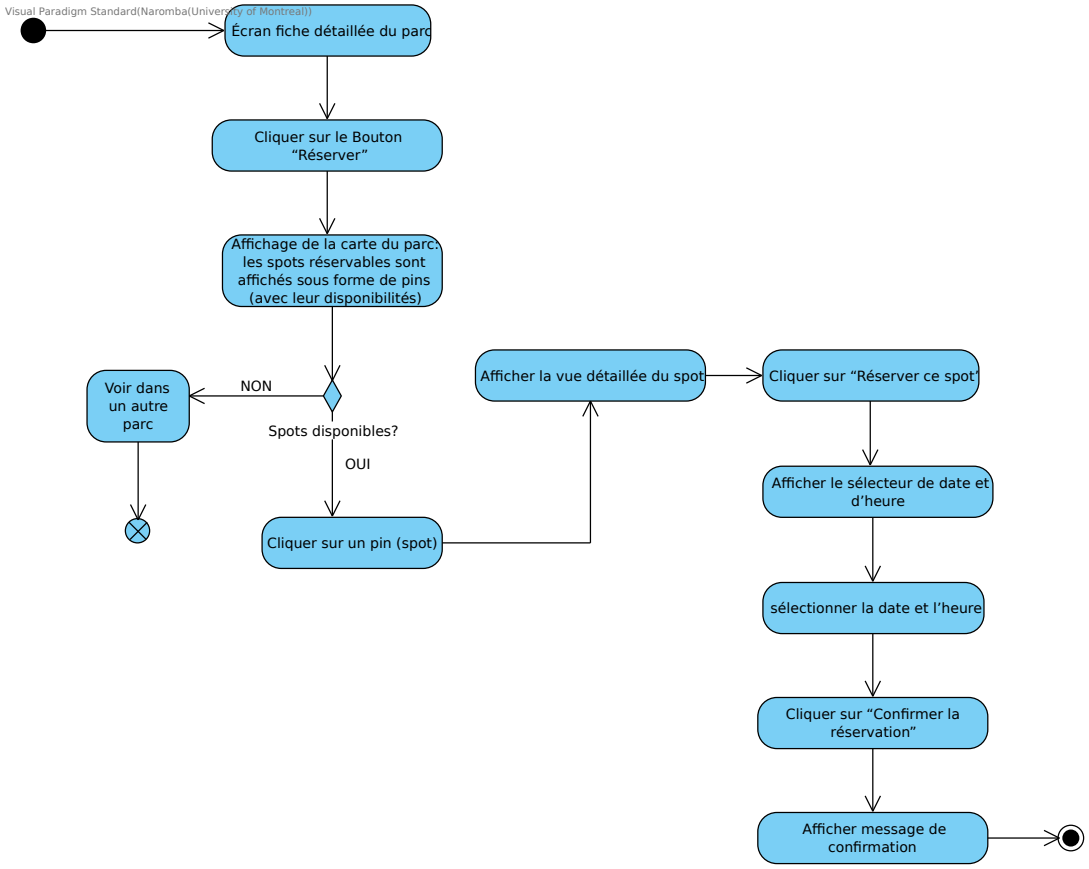

FIGURE 5 – Flux : Réservation d'un emplacement

3.4.4 Réservation d'une activité

Étapes pour sélectionner une activité dans la fiche du parc, choisir un créneau horaire et confirmer la réservation.

3.4.5 Ajout ou retrait d'un favori

Deux cas sont couverts : retirer un parc des favoris depuis la fiche du parc ou depuis la page des favoris.

3.4.6 Ajout d'un avis

L'utilisateur descend jusqu'à la section « Avis », saisit un commentaire, ajoute éventuellement une image, puis envoie son avis.

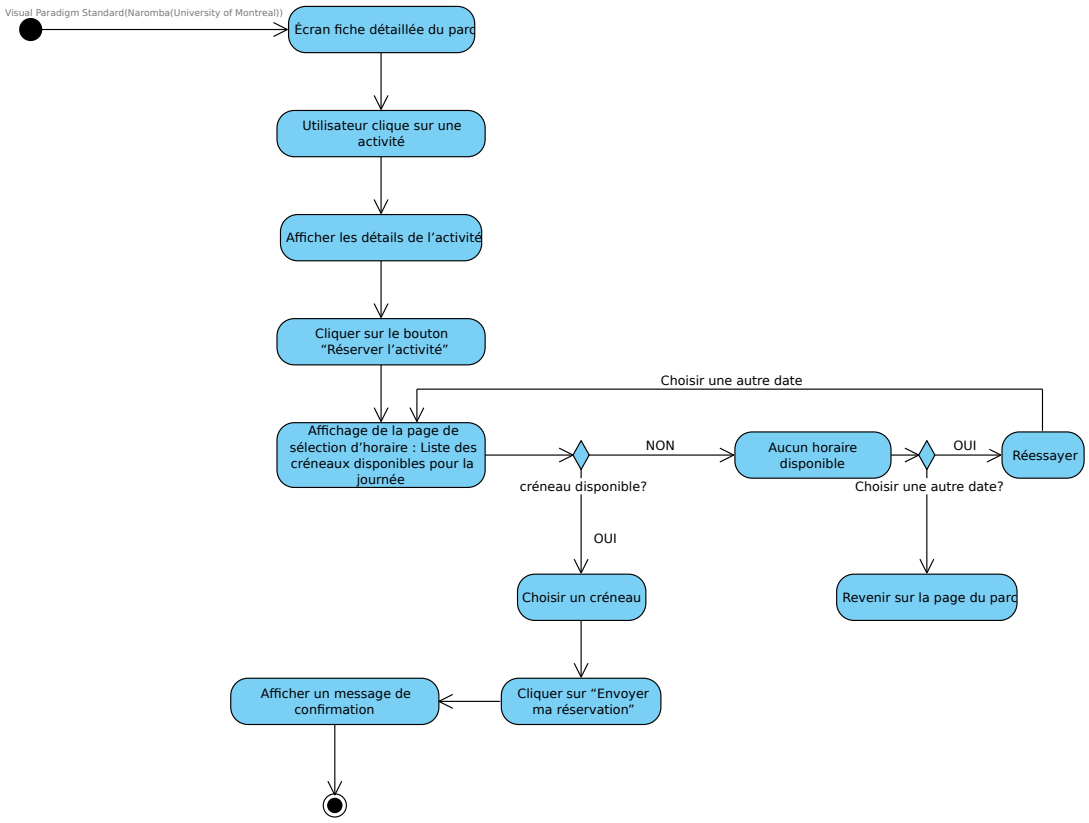

FIGURE 6 – Flux : Réservation d'une activité

3.4.7 Sondage post-visite

Après la visite, l'utilisateur reçoit une notification « Would you Pick-Me again ? », sélectionne des badges et peut laisser un commentaire.

3.5 Prototypes et maquettes

Le design de l'application a été pensé pour offrir une expérience intuitive, visuelle et cohérente sur mobile. La maquette interactive a été conçue avec Figma.

Lien Figma : https://www.figma.com/design/kP7QEAejfvUB2yia4cleau/Pique-me?node-id=0-1

3.6 Contraintes et considérations

— **Techniques :** limiter le nombre d'appels vers les services externes, garantir une connexion correcte même si le réseau est lent ou instable.

— **Réglementaires :** respecter la vie privée des utilisateurs ; permettre la suppression de leurs données s'ils le demandent.

— **Fonctionnelles :** demander à l'utilisateur de confirmer sa réservation, l'annuler automatiquement s'il ne se présente pas dans l'heure, et éviter qu'un même créneau soit

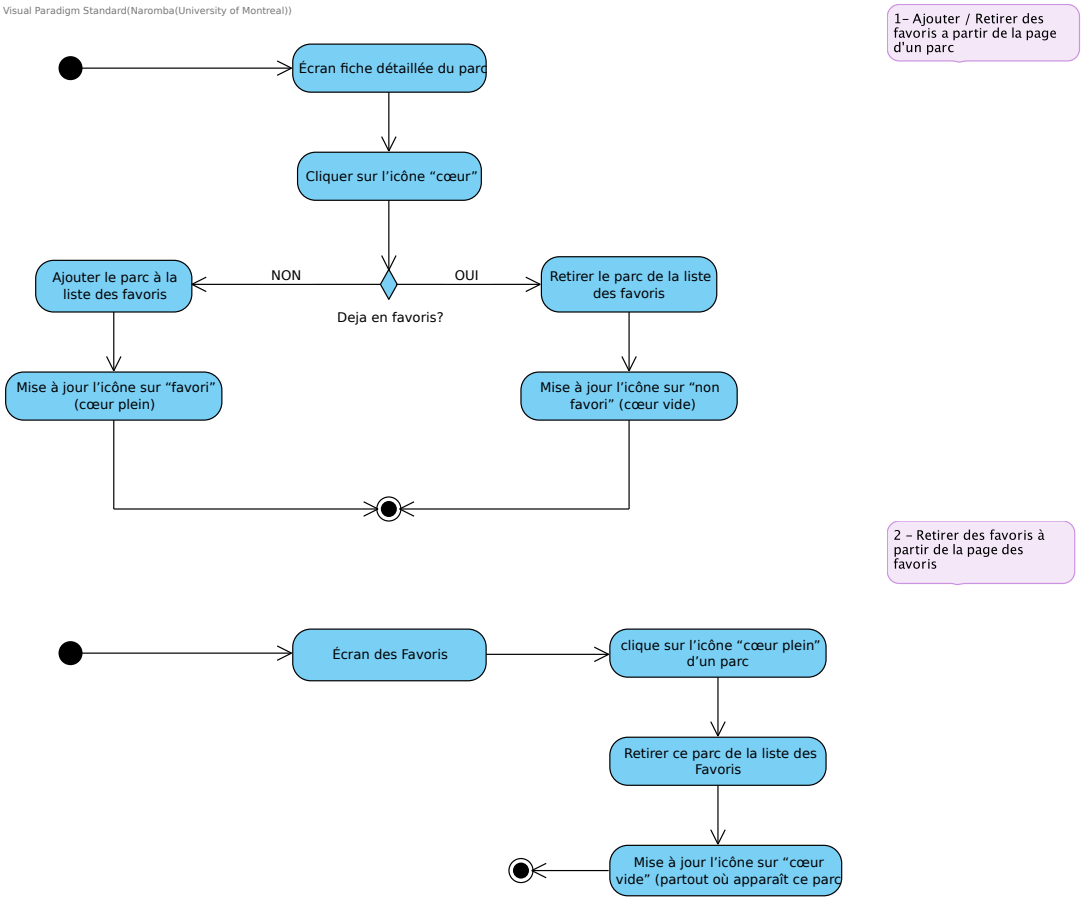

FIGURE 7 – Flux : Ajout / retrait d'un favori

réservé deux fois.

4 Implémentation et réalisation technique

Cette section présente comment nous avons transformé le projet Pique-Me en une vraie application mobile. Après avoir défini les fonctionnalités clés, nous avons découpé l'app en petits composants simples, chacun responsable d'une tâche précise (ex. affichage d'un parc, ajout aux favoris, formulaire de réservation...). En cours de route, nous avons ajusté la conception. Par exemple, l'API de Montréal ne répondait pas entièrement à nos besoins : nous avons donc fusionné plusieurs sources de données, puis stocké le tout dans Firebase. Chaque page suit un flux logique clair, et les données s'actualisent en temps réel grâce aux outils fournis par Firebase.

— L'**interface mobile**, développée avec React Native et Expo, affiche toutes les pages de l'application. Elle gère les gestes de l'utilisateur (clics, défilement, navigation...).

— La **base de données en ligne**, hébergée sur Firebase, contient les informations essentielles : comptes des utilisateurs, réservations d'activités ou d'emplacements, parcs favoris et avis. Elle permet aussi de recevoir les mises à jour instantanément sans avoir

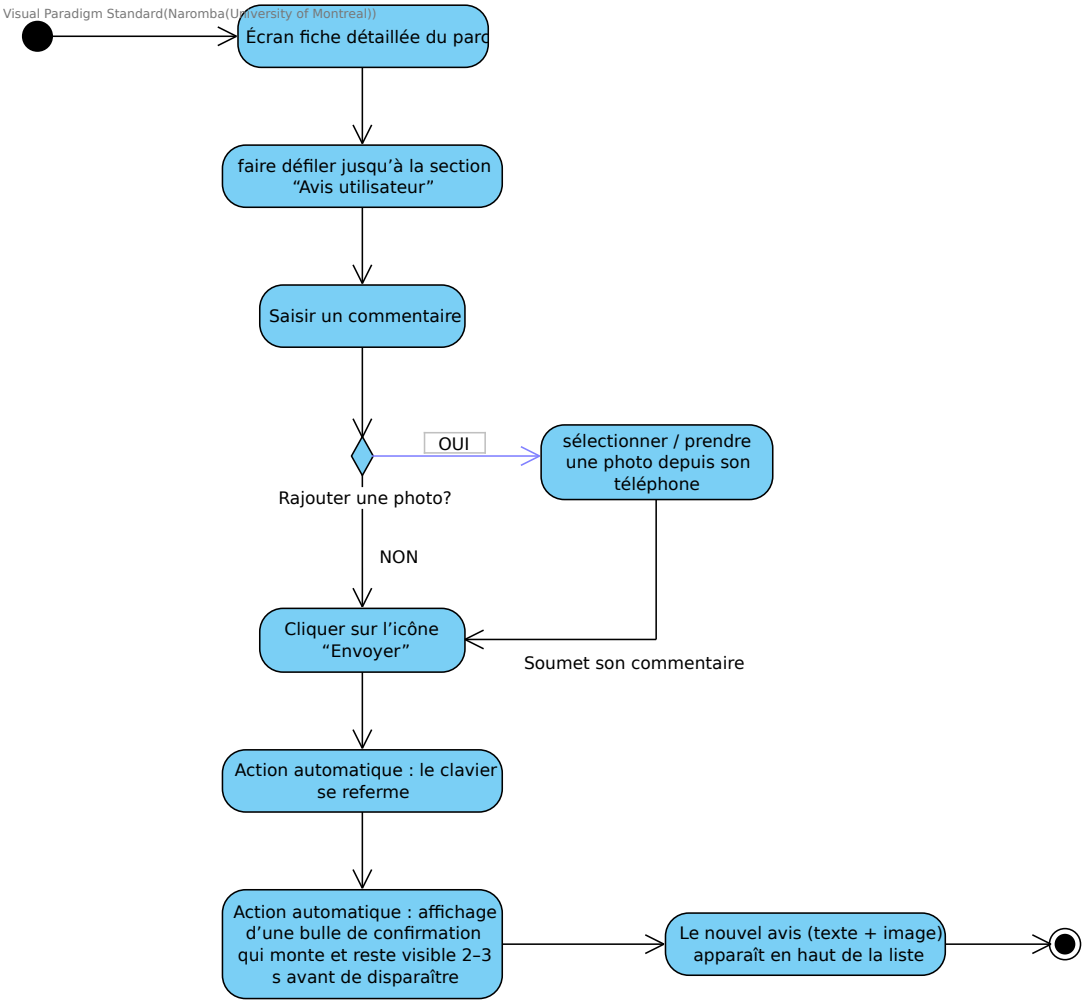

FIGURE 8 – Flux : Ajout d'un avis

à rafraîchir manuellement.

— Un **script d'importation** a été conçu pour récupérer plusieurs ensembles de données ouvertes (emplacements des parcs, installations sportives, règles d'usage). Ces fichiers sont fusionnés, nettoyés et organisés dans une base unifiée, appelée `parks`, utilisée dans toute l'application.

Cette architecture simple évite la gestion d'un serveur complexe : tout est centralisé dans Firebase, ce qui accélère le développement et garantit une bonne réactivité sur mobile.

4.1 Particularités rencontrées

— **Découverte de Firebase :** au début du projet, nous ne connaissions pas Firebase. Il a fallu apprendre à l'utiliser, notamment pour gérer les droits d'accès, les mises à jour en temps réel, l'authentification et la structure des données. Finalement, cela nous a permis de déléguer toute la logique serveur (sécurité, réservations, notifications), ce

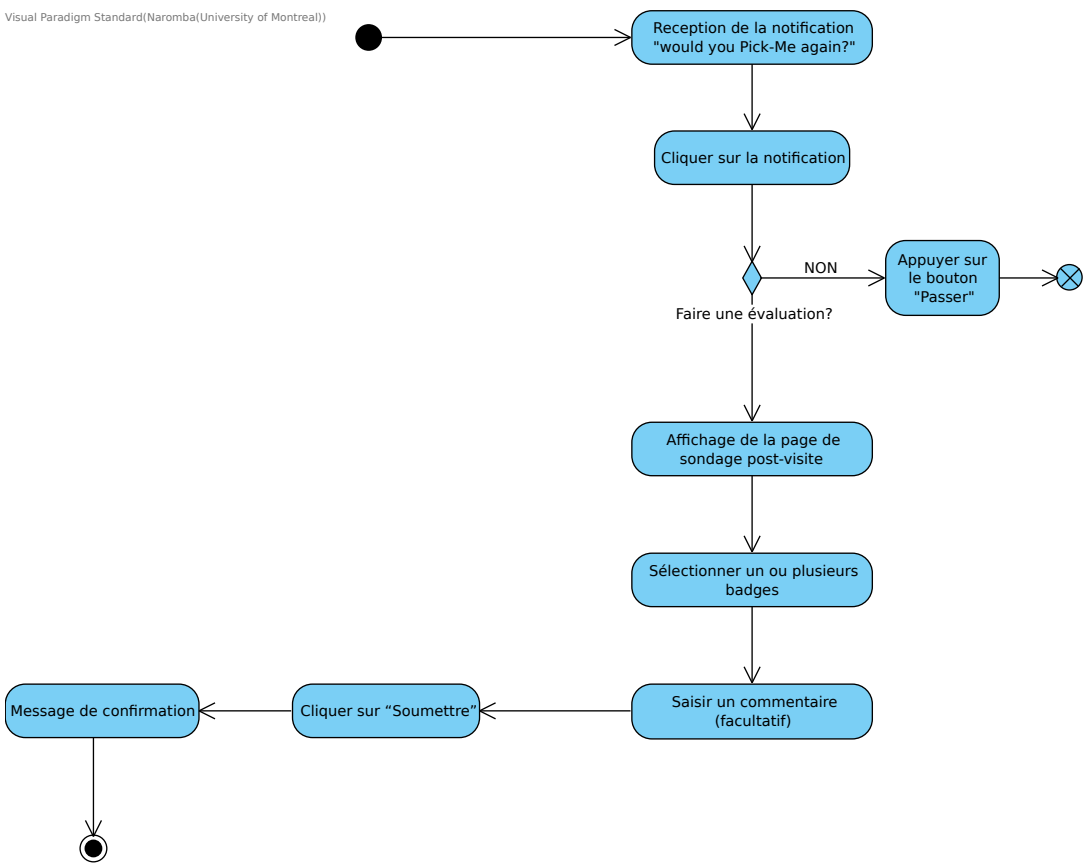

FIGURE 9 – Flux : Sondage post-visite

qui a simplifié considérablement notre code.

— **Gestion correcte des favoris :** pour éviter les doublons ou les incohérences, nous avons conçu une fonction centrale qui gère l'ajout ou la suppression de favoris depuis n'importe quelle page. Ainsi, les états restent synchronisés automatiquement entre l'accueil, la page de favoris et les fiches de parc.

— **Carte et affichage limité des parcs :** pour ne pas saturer visuellement la carte ni ralentir les performances, nous avons limité l'affichage des « pins » aux parcs situés dans un rayon de 2 km autour de l'utilisateur. Cela rend la carte plus fluide et plus lisible, surtout dans les quartiers denses.

— **Réservations concurrentes :** pour éviter que deux utilisateurs réservent le même emplacement au même moment, la base de données vérifie chaque requête en temps réel. Si le créneau est déjà pris, un message d'erreur clair est retourné immédiatement.

4.2 Flux principaux par page

Accueil

Cette page met en avant deux types de parcs : ceux recommandés selon les préférences de l'utilisateur (plein air, sport, famille, etc.), et ceux situés à proximité (sur un rayon

de deux kilomètres à la ronde). Chaque carte de parc peut être cliquée pour accéder à sa fiche détaillée, ou ajoutée directement aux favoris.

Recherche

L'utilisateur peut rechercher un parc par mot-clé ou appliquer des filtres (type de parc, activités disponibles, etc.). **Astuce de vitesse :** au lieu d'attendre que tous les résultats soient prêts, la page affiche immédiatement quelques parcs très pertinents. En parallèle, le reste de la liste continue à se charger discrètement. Cela donne une impression de fluidité et de réactivité.

Détail d'un parc

Chaque fiche de parc présente une image principale, les équipements disponibles ou activités, et deux boutons : « Réserver » pour choisir un créneau ; une icône en forme de cœur pour ajouter ou retirer ce parc de la liste personnelle. L'ensemble est mis à jour en temps réel via Firebase.

Réservation

L'utilisateur sélectionne un emplacement et un créneau horaire. Une confirmation visuelle apparaît. 60 minutes avant l'heure choisie, une notification est envoyée. Si l'utilisateur ne se présente pas, la réservation est annulée automatiquement une heure après le début prévu.

Favoris

Les parcs ajoutés aux favoris sont listés dans une page dédiée. Chaque carte peut être retirée en cliquant sur l'icône cœur. Toute modification est immédiatement répercutée sur les autres pages.

Profil

Cette page permet de modifier le nom, le mot de passe et les préférences (tags). Chaque modification est enregistrée automatiquement dès que l'utilisateur quitte le champ de saisie. Les préférences servent à personnaliser les suggestions de parcs à l'accueil.

Avis

Après une visite, l'utilisateur reçoit une notification l'invitant à donner son avis. Il peut sélectionner un ou plusieurs badges (ex. : propre, calme, familial), laisser un commentaire et ajouter une photo. Si un badge reçoit beaucoup de votes, le parc obtient une médaille (bronze, argent ou or), visible dans sa fiche.

4.3 Difficultés résolues

— **Fusion de données publiques :** les informations sur les parcs étaient réparties dans plusieurs fichiers (localisation, équipements et activités). Nous avons dû croiser et nettoyer ces données pour n'avoir qu'un seul format clair à utiliser dans l'application.

— **Interrogation des API trop lente :** certaines sources prenaient jusqu'à deux minutes pour répondre. Nous avons donc décidé d'extraire leurs données à l'avance et de les importer une fois pour toutes dans Firebase.

— **Problèmes de réservation en double :** pour éviter que deux utilisateurs réservent le même créneau, nous avons utilisé une vérification automatique côté base de données. Si le créneau est déjà pris, un message d'erreur clair s'affiche.

— **Affichage pas toujours à jour :** au début, il arrivait que les favoris ou les avis ne se mettent pas à jour tout de suite. Nous avons corrigé cela en connectant directement les composants d'affichage à Firebase. Ainsi, tout changement est visible en temps réel, sans avoir besoin de recharger l'écran.

5 Évaluation

5.1 Tests unitaires et fonctionnels

5.2 Évaluation de performance

5.3 Évaluation d'utilisabilité

5.4 Comparaison avec les objectifs initiaux

6 Discussion critique

La version actuelle de *Pique-Me* atteint son objectif premier : offrir, sur mobile, un moyen simple de découvrir les parcs de Montréal, de réserver un emplacement et de partager son expérience.

Analyse critique. Le choix d'un socle « Firebase » a apporté un réel confort : pas de serveur à maintenir, sécurité intégrée, mises à jour en temps réel gratuites. En revanche, cette dépendance impose certaines limites : logique métier cantonnée aux règles Firestore, format de requête parfois rigide, et difficulté à exécuter des traitements lourds côté serveur. La structure en mini-composants React facilite grandement la maintenance ; toutefois, elle augmente aussi le nombre de fichiers à gérer et peut alourdir la prise en main pour un nouveau développeur.

Problèmes rencontrés et solutions mises en place.

— *Données éparpillées :* les informations officielles provenaient de trois sources lentes et incomplètes. Nous avons donc écrit un script d'import qui fusionne et nettoie ces fichiers, puis les stocke dans une seule collection `parks`. Temps de réponse côté appli : divisé par dix.

— *API cartographique saturée :* afficher 500 parcs à la fois rendait la carte illisible. Nous limitons maintenant les marqueurs au rayon de 2 km ; les autres parcs se chargent au besoin quand on dézoome.

— *Réservations en double :* deux utilisateurs pouvaient, durant les premiers tests, choisir le même créneau. Désormais, une transaction Firestore bloque la seconde requête et renvoie un message explicite.

— *Synchronisation des favoris :* il arrivait qu'un cœur reste allumé sur un écran mais pas sur l'autre. Tous les composants « carte de parc » écoutent maintenant directement les changements Firebase ; l'affichage est harmonisé sans rechargement manuel.

Ce qui pourrait être amélioré avec plus de temps ou de moyens.

— *Suggestions communautaires :* permettre aux utilisateurs d'ajouter eux-mêmes de nouveaux parcs ou équipements encore non répertoriés, photos à l'appui.

— *Recommandations plus fines :* analyser les avis pour proposer des parcs en fonction du moment de la journée, de la météo ou d'événements spéciaux.

— *Plateformes supplémentaires :* décliner l'application en version tablette et web pour toucher un public plus large : familles planifiant leurs sorties depuis un salon ou écoles préparant des activités extérieures.

— *Fonctions hors ligne étendues :* permettre le téléchargement d'un arrondissement complet afin d'utiliser la recherche cartographique sans réseau du tout.

— *Interface adaptable :* proposer un mode « gaucher » (boutons principaux inversés) et des options d'accessibilité supplémentaires (taille de police, contraste renforcé).

— *Multilingue :* offrir l'application en français et en anglais (et d'autres langues à terme) afin d'être inclusive pour les résidents et touristes.

7 Conclusion

Le projet *Pique-Me* visait à simplifier l'expérience des citoyens dans les parcs urbains : trouver le bon lieu, réserver un espace et partager un avis. En quatre mois, nous avons livré une application mobile fluide, capable d'afficher les parcs pertinents en quelques secondes et de synchroniser réservations ou favoris en temps réel.

Leçons tirées. Nous avons mesuré la force et les limites d'une solution entièrement basée sur Firebase : elle accélère la mise en œuvre et la synchronisation, mais impose un

format de requête strict. L'approche « mini-composants » sous React Native s'est révélée payante : chaque pièce est réutilisable et testable isolément, limitant les régressions. Enfin, fusionner plusieurs jeux de données publics nous a appris qu'un nettoyage rigoureux avant importation est indispensable pour garantir une source unique et fiable.

Pistes d'amélioration.

— *Fonctions communautaires élargies :* permettre aux usagers de proposer de nouveaux parcs ou équipements, photos à l'appui.

— *Recommandations personnalisées :* intégrer la météo, l'heure de la journée et les avis pour suggérer le parc idéal à chaque instant.

— *Accessibilité renforcée :* mode gaucher, tailles de texte adaptatives, synthèse vocale pour la navigation.

— *Version web et tablette :* offrir la même expérience sur grand écran, utile aux familles et aux écoles.

— *Mode hors-ligne complet :* téléchargement d'un arrondissement pour une consultation sans réseau, puis synchronisation différée.

En définitive, *Pique-Me* prouve qu'une architecture simple et modulaire peut rendre un service concret et apprécié. Avec ces améliorations, nous pourrons **étendre la couverture et faire de *Pique-Me* l'outil de référence des sorties dans les parcs montréalais**.

Remerciements

Nous souhaitons exprimer notre gratitude envers nos collègues **Marc Oliver Jean paul** et **Lallia Diakité**, avec qui nous avons partagé le développement de ce projet dans le cadre du cours *Interface personne-machine*. Leur collaboration, leur écoute et leur motivation ont grandement contribué à la réussite de ce travail.

Nous remercions également **M. Louis-Édouard Lafontant**, notre enseignant, pour son accompagnement tout au long de la session. Présent à chaque étape, il a su répondre à toutes nos questions avec rigueur et bienveillance, parfois dans le cadre de discussions (très) longues mais toujours constructives. Il n'a pas hésité à nous faire retravailler plusieurs éléments pour atteindre un meilleur rendu.

Enfin, nous tenons à saluer **notre propre engagement** : malgré une session intense, nous avons su rester investis, rigoureux et persévérants, jusqu'à la finalisation de ce projet.

Références

— Lafontant, L.-É. *Supervision, encadrement et conseils.*

— Google. (2024). *Documentation Firebase*. Disponible en ligne : https://firebase.google.com/docs

— Ville de Montréal. (2024). *API publiques — données sur les parcs, équipements et activités. (....)*.

— Meta. (2024). *Documentation React Native*. Composants de base, navigation et gestion d'état. *(....)*.

— OpenAI. (2025). *Assistance à la rédaction et au débogage via ChatGPT*. https://chat.openai.com

A Annexes