Rapport projet d'intégration

William Wauters (PSR10589)

Professeur chargé d'encadrement : Jean-Paul HECQUET

TABLE DES MATIÈRES

Installation du projet

Vous trouverez le code de l'application sur mon GitHub (https://github.com/WilliamGordon/GeoCity)

FRONT-END - REACT JS

1. Installer Node (version 16.13.0)
2. Installer les packages avec la commande « npm install »
2. Lancer l'application avec la commande « npm start »

BACK-END - API .NET 6

1. Installer . NET 6 SDK
2. Installer Visual Studio 2022
3. Lancer la solution "geocity.api.sln"
4. Créer le fichier « appsettings.json » dans le projet ASP.NET API « geocity.api »
5. Configurer le fichier « appsettings.json » avec la « connection string » de la base de données
6. Configurer le fichier « appsettings.json » avec les informations d'auth0
3. Lancer l'application avec « IIS Express » sur Visual Studio 2022

DATABASE – SQL SERVER

1. Installer une instance d'SQL Server
2. Créer la base de donnée en utilisant le script « GeoCity.sql » dans le dossier Geocity > geocity.db
   Ou
2. Créer la base de données GeoCity et utilisé la commande « Update-Database » avec le Package Manager Console dans VS2022 avec comme projet par défaut « geocity.infrastructure »

1. Introduction

Ce travail décrit le processus d'analyse et de développement d'une application permettant à des utilisateurs de créer des citytrips.

Pour ce faire, nous tenterons avec ce projet d'utiliser des technologies qui offrent la possibilité d'inclure des cartes interactives. Ainsi, il s'agira de choisir et d'implémenter différents services qui nous donneront les moyens de répondre aux attentes des utilisateurs.

Notre objectif est de délivrer une application robuste et performante capable d'offrir des interfaces utilisateurs riches et innovantes pour permettre aux utilisateurs de facilement créer et organiser des citytrips.

Nous commencerons par définir les contours du projet en décrivant les différentes fonctionnalités devant être implémentées. Par la suite, nous analyserons et formaliserons les processus qui prennent part dans notre application à l'aide de plusieurs diagrammes et schémas. Nous continuerons en créant une base de données robuste basée sur cette analyse. Enfin, nous terminerons en présentant la phase de développement et les différentes technologies utilisées pour mener à bien ce projet.

2. Cahier des charges

Description de la solution envisagée

GeoCity est une suite d'applications permettant de créer et consulter des itinéraires pour la visite d'une ville ou l'organisation d'un citytrip.

Après la création d'un compte, l'utilisateur aura la possibilité de créer un city citytrip. En fonction du nombre de jours, l'utilisateur pourra ensuite créer un itinéraire pour chaque jour. À l'aide d'une carte interactive, l'utilisateur choisit différents points d'intérêts et étapes pour un itinéraire. L'application permet ensuite de calculer la route optimale en fonction des étapes sélectionnées.

Lors de la configuration de l'itinéraire, une liste de suggestion d'étapes populaires sera proposée aux utilisateurs. Cette liste sera composée en fonction des points d'intérêt précédemment sélectionnés par les autres utilisateurs de l'application.

Un utilisateur peut inviter un autre utilisateur à participer à la création du citytrip. Le créateur du citytrip peut ensuite décider de publier son citytrip publiquement.

Les citytrips publiés seront accessibles à tous les utilisateurs. Les utilisateurs pourront ainsi accéder à une liste de citytrip qu'il sera possible de trier par ville, distance, nombre de jours, popularité. Les utilisateurs pourront ajouter des citytrips dans leur liste de favoris.

Un système de rating sera mis en place ou chaque utilisateur aura l'occasion de donner une note pour un itinéraire publié.

Les itinéraires créés sont prévus pour être faits exclusivement à pied. En effet, l'intention est de découvrir une ville en la parcourant à pied. GeoCity donne la possibilité de créer des itinéraires touristiques avec une description détaillés des différentes étapes, ou plus simplement des promenades et ballades pour découvrir un quartier spécifique.

Le projet comportera 2 applications. La première permet de créer et configurer des citytrips. La deuxième permet seulement de consulter les citytrips publiés, ou ses propres citytrips. Les utilisateurs pourront ainsi en premier lieu créer et organiser leur citytrip avec une application Web, et par la suite, les consulter avec l'application mobile afin de faciliter leur voyage.

- **Application Web** : Cette application permettra aux utilisateurs de créer et configurer des citytrips. Il sera possible pour les utilisateurs d'ajouter des étapes, de générer des itinéraires, de publier et partager leurs citytrips avec d'autres utilisateurs. Il sera également possible de consulter les citytrips qui ont été rendus publics par leurs auteurs.

- **Application mobile** : Cette application permettra aux utilisateurs de consulter leurs citytrips.

Intervenants

Plusieurs types d'intervenants seront rencontrés dans l'application. Ces utilisateurs sont différenciés en fonction de leur relation avec un citytrip.

- *Visiteur* : Cet intervenant ne possède pas un compte utilisateur. Le visiteur peut accéder aux fonctionnalités ne nécessitant pas de connexion.

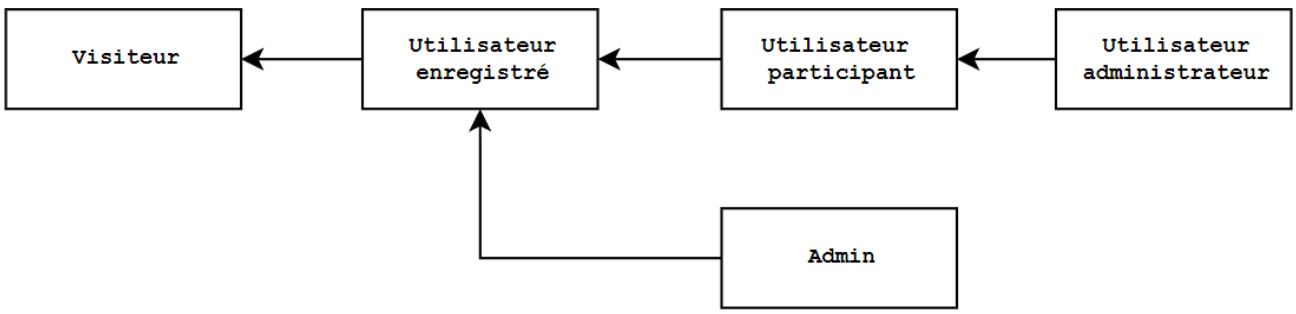
- *Utilisateur enregistré* : Un utilisateur enregistré est un utilisateur qui a préalablement créé un compte utilisateur et est identifié comme tel. Cet utilisateur a les mêmes privilèges qu'un visiteur. Il lui est donné la possibilité de créer son propre citytrip. Il peut également ajouter des citytrips dans ses favoris et donner une note à des citytrips publiés.

- *Utilisateur participant* : Un utilisateur participant a les mêmes privilèges qu'un utilisateur enregistré. Cet utilisateur est inscrit comme participant à un citytrip préalablement créé. Il a le droit d'ajouter, modifier, ou supprimer des étapes.

- *Utilisateur administrateur*: Un utilisateur administrateur a les mêmes privilèges qu'un utilisateur participant. Il est le propriétaire du citytrip qu'il a créé. Il lui est donné la possibilité de publier son citytrip afin de le rendre visible pour les utilisateurs enregistrés. Il peut également inviter des utilisateurs à participer son citytrip.

- Administrateur : Il peut accéder à la section de gestion des étapes suggérées afin de contrôler les étapes pouvant être recommandées par des utilisateurs à d'autres utilisateurs. Il a également accès à la liste des utilisateurs.

Les différents intervenants s'inscrivent dans une relation d'héritage. Les droits et permissions par rapport à un citytrip sont hérités en fonction de leur relation par rapport au citytrip. Les visiteurs et utilisateurs enregistrés ont seulement un droit de lecture des citytrips publiés. Les utilisateurs participants ont les droits d'édition des étapes d'un citytrip pour lequel ils sont inscrits. Les utilisateurs administrateurs ont les droits d'édition et les droits de publication d'un citytrip.



Fonctionnalités

Exigences fonctionnelles

Nous retrouverons ici un tableau récapitulatif des différentes fonctionnalités identifiées pour l'application GeoCity.

Fonctionnalités visiteur		
Code	***Description***	***Application***
EF-001	Créer un compte	Web & Mobile
EF-002	Effectuer une recherche	Web & Mobile
EF-003	Afficher les détails d'un citytrip	Web & Mobile
EF-004	Afficher les détails d'un point d'intérêt	Web & Mobile

Fonctionnalités utilisateur enregistré		
Code	***Description***	***Application***
EF-004	Se connecter	Web & Mobile
EF-005	Noter un citytrip	Web & Mobile
EF-006	Ajouter un citytrip aux favoris	Web & Mobile
EF-007	Afficher ses citytrips	Web & Mobile
EF-008	Créer un citytrip	Web
EF-009	S'inscrire à un citytrip	Web

Fonctionnalités utilisateur participant		
Code	***Description***	***Application***
EF-010	Afficher les détails d'un citytrip	Web
EF-011	Ajouter une étape	Web
EF-012	Modifier un point d'intérêt	Web
EF-013	Supprimer une étape	Web
EF-014	Générer un itinéraire	Web

Fonctionnalités Utilisateur administrateur		
Code	***Description***	***Application***
EF-015	Modifier un citytrip	Web
EF-016	Supprimer un citytrip	Web
EF-017	Publier un citytrip	Web
EF-018	Partager un citytrip	Web

Fonctionnalités Administrateur		
Code	***Description***	***Application***
EF-019	Afficher les utilisateurs	Web
EF-020	Afficher les détails d'un utilisateur	Web
EF-021	Afficher les détails d'une ville	Web
EF-022	Afficher les suggestions d'une ville	Web
EF-023	Ajouter une suggestion	Web

Identification, Authentification et autorisation

Les visiteurs ont la possibilité de créer un compte. Nous utiliserons dans nos applications les services d'Auth0 pour l'identification et l'authentification des utilisateurs. Les utilisateurs auront également la possibilité d'utiliser un compte Google préexistant.

Le processus d'identification et d'authentification sera identique pour l'application Web et l'application mobile.

Nous utiliserons également Auth0 pour différencier les administrateurs des autres utilisateurs. Néanmoins, la gestion des autorisations pour les citytrips se fera en interne pour pouvoir différencier un utilisateur participant d'un utilisateur administrateur en fonction d'un citytrip particulier.

Gestion des citytrips

Effectuer une recherche

Les visiteurs peuvent effectuer des recherches paramétrées afin de trouver les citytrips qui les intéresseraient. Dans un premier temps, une recherche simple peut être effectuée en encodant le nom d'une ville. Toutefois, l'application propose également aux utilisateurs une série de critères de recherche pour affiner les résultats de leurs recherches. Nous retrouverons cette fonctionnalité dans les deux applications. On retrouve les critères de recherche suivants :

- **Ville** : représente la ville sélectionnée et concernée par la recherche. (champs obligatoires)
- **Le nombre de jours** : représente le nombre de jours d'un citytrip
- **Le budget** : représente le budget total d'un citytrip
- **La note** : représente la note globale calculée en fonction des notes données par les utilisateurs

Afficher les détails d'un citytrip

Chaque citytrip peut être affiché en détail pour permettre aux utilisateurs de parcourir les différents itinéraires et étapes qui les composent. Cette page de détails contient l'ensemble des informations concernant un citytrip :

- **Détails du citytrip :** Le nom, la description, le nombre de jours, la note du citytrip
- **L'auteur :** Nom de l'auteur et date de publication
- **Détails des étapes :** Le premier jour du citytrip est affiché par défaut. L'utilisateur peut ensuite naviguer vers les différents itinéraires et les étapes qui les composent.
- **Carte interactive** : Une carte interactive où l'on retrouve l'itinéraire dessiné pour le jour sélectionné.

L'affichage des informations sera différent en fonctions des applications Web et mobiles pour rendre un visionnage agréable pour chaque interface.

Créer, modifier et supprimer un citytrip

La création d'un citytrip et sa configuration constitue le cœur de l'application. Les utilisateurs auront l'opportunité de pouvoir créer leur propre citytrip en choisissant une ville, un nom et un nombre de jours (maximum 5). Une description optionnelle peut également être donnée.

La création d'un citytrip implique plusieurs opérations sous-jacentes. En effet, un citytrip est composé de plusieurs itinéraires représentant chaque jour du citytrip. Les itinéraires sont à leur tour composés de différentes étapes. Lorsqu'un utilisateur crée un citytrip, l'application crée automatiquement un itinéraire pour chaque jour sélectionné.

Après la création du citytrip et des itinéraires, l'utilisateur sera invité à ajouter des étapes pour chaque itinéraire. Ceci pourra être fait de plusieurs manières :

- **« Point and click »** : Un clique sur la carte interactive permet d'ajouter une étape.
- **Barre de recherche** : Au cas, où l'utilisateur ne sait pas où se trouve une étape sur la carte, il peut effectuer une recherche à l'aide d'une barre de recherche.
- **Suggestions** : une liste de suggestion d'étapes populaires sera proposée aux utilisateurs. Cette liste sera composée en fonction des étapes précédemment sélectionnées par les autres utilisateurs de l'application.

Lors de la création d'une étape, l'utilisateur devra différencier si cette étape est un point de passage ou un point d'intérêt. Si l'étape ajoutée est un point de passage, aucune autre information n'est demandée. À l'inverse, si l'étape est un point d'intérêt, l'utilisateur pourra ajouter un prix, une durée et une description. Le nom du point d'intérêt sera automatiquement récupéré par le biais d'une API.

Partager un citytrip et s'inscrire à un citytrip

L'utilisateur créateur du citytrip peut partager son citytrip avec d'autres utilisateurs. Pour ce faire, il suffira à l'utilisateur administrateur du citytrip de transmettre à un autre utilisateur un lien unique. La transmission de cette URL n'est pas gérée par l'application. C'est l'utilisateur administrateur du citytrip qui doit de lui-même récupérer ce lien unique et le faire parvenir à d'autres utilisateurs par ses propres moyens. Nous suivons ici le même modèle du partage de ressources utilisé par Google. Il suffit d'avoir un compte et une URL valide pour pouvoir ajouter un fichier Excel, Drive, ou autre.

De l'autre côté, il suffira à un utilisateur enregistré de visiter l'URL pour être automatiquement inscrit comme participant au citytrip. Dans le cas d'un utilisateur n'ayant pas encore un compte, il sera en premier lieu redirigé vers un écran de création d'un compte. Pour éviter d'avoir trop d'utilisateurs pour un seul citytrip, l'application ne permet qu'un maximum de 5 utilisateurs participants pour un seul citytrip.

Publier un citytrip

L'utilisateur administrateur du citytrip à la possibilité de publier son citytrip afin de le rendre accessible par les autres utilisateurs. Avant de publier son trip, un utilisateur administrateur devra confirmer son intention.

Gestion des suggestions

Les administrateurs sont les seuls à avoir accès à la gestion des suggestions. Les suggestions sont toutes les étapes des citytrips qui ont été publiés et qui ont été ajoutés par au moins 5 utilisateurs différents.

L'utilisateur administrateur pourra consulter les points d'intérêt en fonction d'une ville. Il pourra également ajouter un point d'intérêt pour une ville et directement lui donner le statut de suggestion.

Seules les étapes de type point d'intérêt peuvent être des suggestions.

Exigences non fonctionnelles

- L'application doit être disponible tout le temps
- L'interface de configuration des citytrips doit être efficace et fluide pour les utilisateurs.

Architecture du projet

Comme énoncé précédemment, notre solution comprendra deux applications clientes distinctes. Une application Web de type Single Page Application avec le Framework ReactJS, ainsi qu'une application mobile développée en Flutter. Ces deux applications clientes utiliseront des services spécialisés dans l'inclusion de cartes interactives.

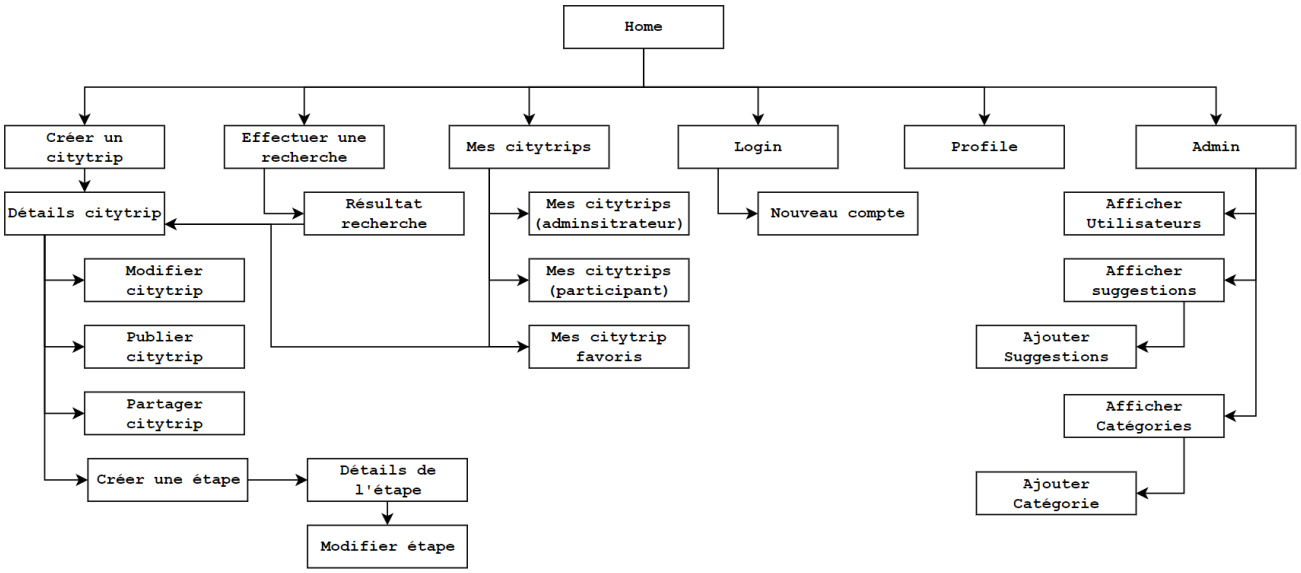
De côté serveur, nous retrouvons une API développée en .NET qui implémentera le « design pattern » CQRS. Notre API sera reliée à notre base de données SQL Server à l'aide d'Entity Framework.

3. Analyse fonctionnelle

Diagramme de navigation des écrans

Application Web

On trouvera dans le schéma suivant la disposition des différentes pages qui composent l'application Web.



Application mobile

Ci-dessous, nous retrouvons le schéma de navigation de notre application mobile. Ce dernier est semblable à celui de l'application Web pour garder une homogénéité entre les deux expériences utilisateurs.

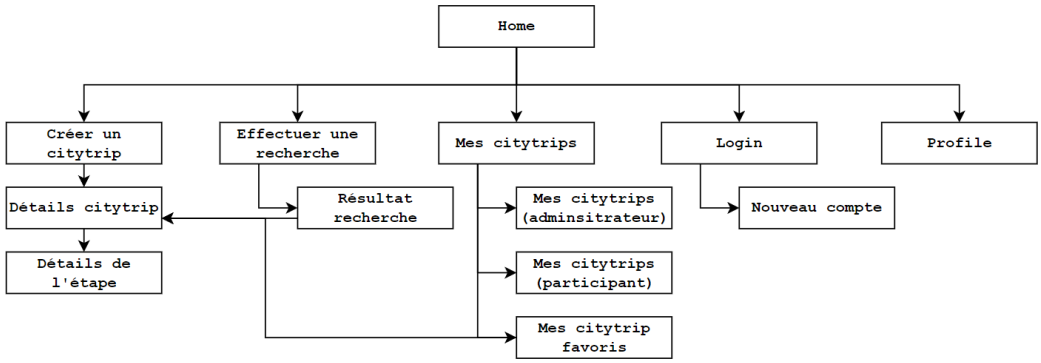
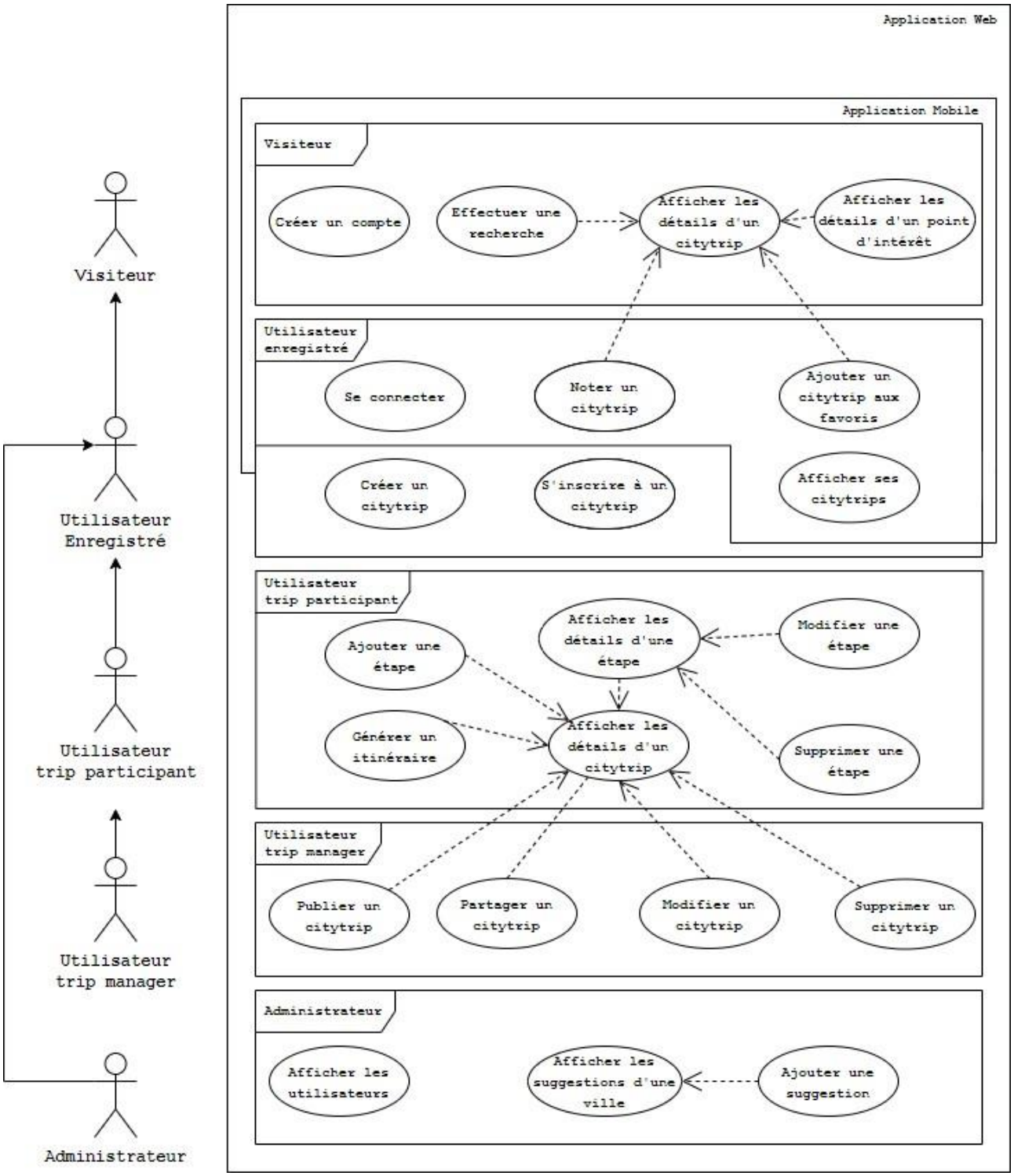


Diagramme des use cases

Diagramme des uses cases



Description des uses cases

Use cases pour l'acteur visiteur

Créer un compte

Titre : Créer un compte
Résumé : Le visiteur procède à la création d'un compte.
Acteurs : Visiteur, système

Préconditions :

- Le visiteur n'a pas un compte

Scénario nominal :

1. Le visiteur accède à l'application Web ou mobile.
2. Le visiteur navigue vers la section « Login ».
3. Le système présente un formulaire d'authentification au visiteur.
4. Le visiteur clique sur l'onglet « Créer un compte ».
5. Le visiteur encode son adresse email, son mot de passe et clique sur le bouton « Envoyer ».
6. Le système enregistre le visiteur en base de données.
7. Le système redirige le visiteur vers l'écran d'accueil.

Enchainements alternatifs :

A1. Le visiteur choisit l'option de s'authentifier avec un compte Google préexistant.
Démarre au point 3 du scénario nominal.
4. Le visiteur clique sur l'option « Se loguer avec Google ».
5. Le système redirige le visiteur vers l'écran d'accueil.

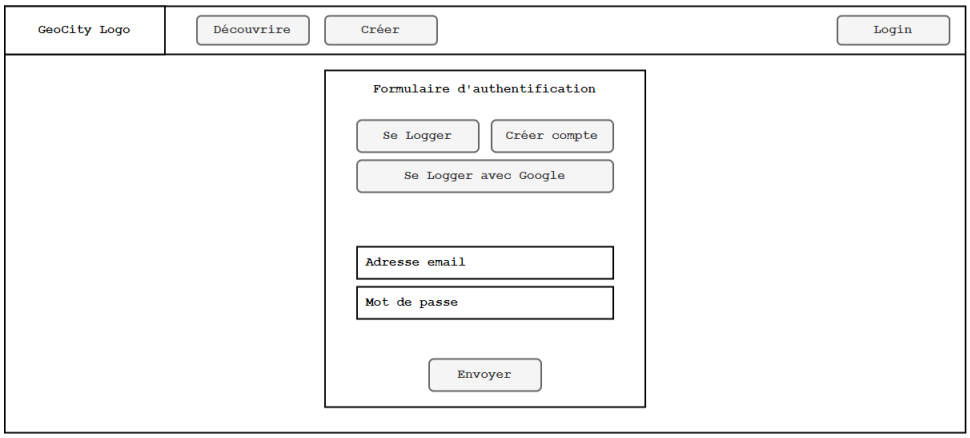
Enchainement d'erreurs :

E1. Le visiteur encode des données erronées.
Démarre au point 5 du scénario nominal.
6. Le système présente au visiteur les erreurs de validation.
Le scénario nominal reprend au point 5.

Post-conditions :

- Le visiteur est enregistré en base de données.
- Le visiteur devient un utilisateur enregistré.

Visuel/Écran :


GeoCity Logo
Découvrire
Créer
Login
Formulaire d'authentification
Se Logger
Créer compte
Se Logger avec Google
Adresse email
Mot de passe
Envoyer


GeoCity Logo
Menu
Se Logger
Créer compte
Se Logger avec Google
Adresse email
Mot de passe
Envoyer

Effectuer une recherche

Titre : Effectuer une recherche
Résumé : Le visiteur peut rechercher un citytrip en fonction de critères spécifiques.
Acteurs : Visiteur, système

Scénario nominal :

1. Le visiteur accède à l'application Web ou mobile.
2. Le visiteur navigue vers la section « découvrir ».
3. Le visiteur encode les critères de recherche et lance la recherche.
4. Le système récupère les citytrips correspondants aux critères de recherche.
5. Le système affiche les résultats de la recherche.

Enchainements alternatifs :

A1. Le visiteur choisit l'option « critères de recherche avancée » pour affiner sa recherche.
Démarre au point 3 du scénario nominal.
4. Le visiteur encode les critères de recherche avancée et lance la recherche.
Le scénario nominal reprend au point 4.

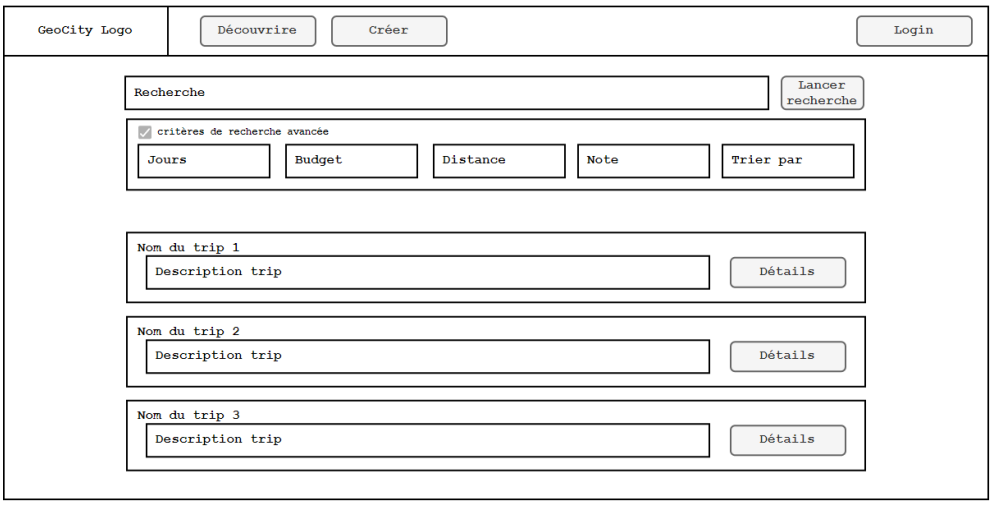
Enchainement d'erreurs :

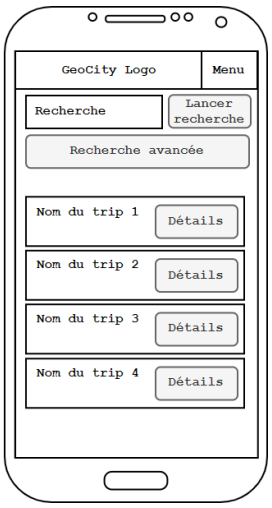
E1. Le visiteur encode des données erronées.
Démarre au point 3 du scénario nominal.
4. Le système présente au visiteur les erreurs de validation.
Le scénario nominal reprend au point 3.

Post-conditions :

- La liste des citytrips est affichée en fonctions de critères de recherche.

Visuel/Écran :





Afficher les détails d’un citytrip

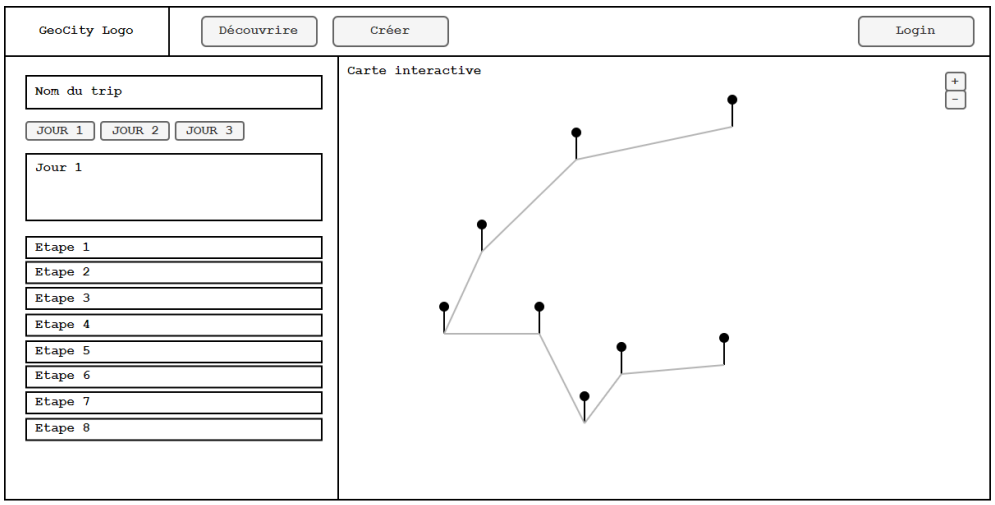
Titre : Afficher les détails d’un citytrip
Résumé : Le visiteur peut consulter en détail un citytrip.
Acteurs : Visiteur, Système

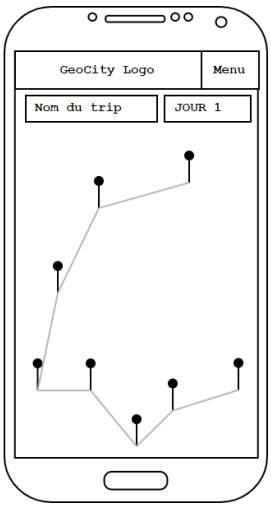
Préconditions :
- Effectuer une recherche

Scénario nominal :
1. Le visiteur clique sur le bouton « détails » sur la fiche du citytrip.
2. Le système récupère les informations de ce citytrip.
3. Le système redirige le visiteur vers l’écran détails d’un citytrip.

Post-conditions
- Les informations du citytrip sont affichées.

Visuel/Écran :





Afficher les détails d’un point d’intérêt

Titre : Afficher les détails d’un point d’intérêt
Résumé : Le visiteur peut consulter les détails d’un point d’intérêt.
Acteurs : Visiteur, Système

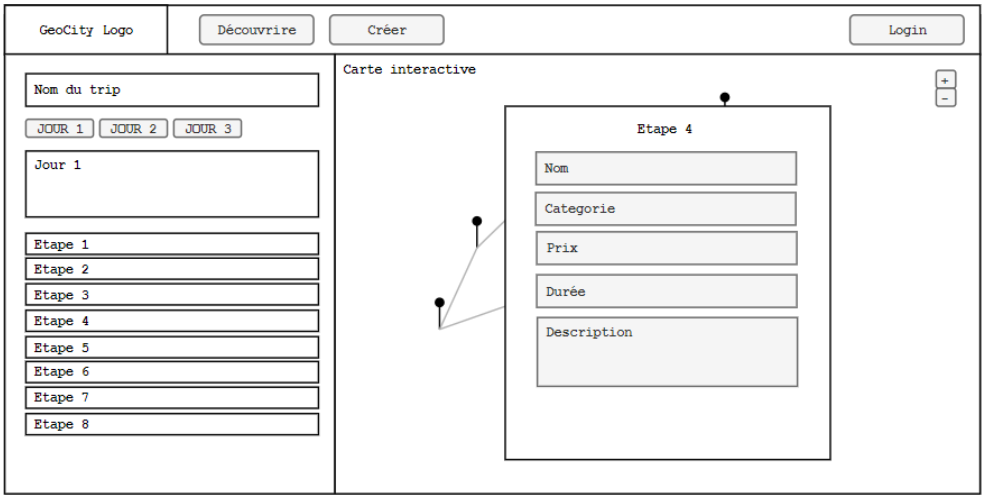
Préconditions :
- Afficher les détails d’un citytrip

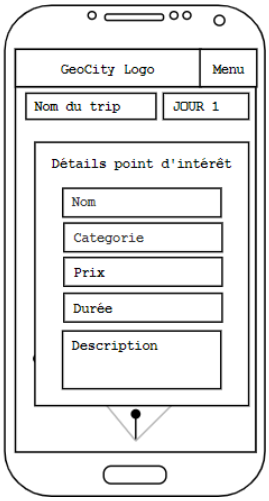
Scénario nominal :
1. Le visiteur clique sur un point d’intérêt dessiné sur la carte.
2. Le système récupère les informations de ce point d’intérêt.
3. Le système présente un écran avec les détails du un point d’intérêt.

Enchainements d’erreurs :
A1. L’utilisateur clique sur une étape de type point de passage.
Démarre au point 2 du scénario nominal.
3. Le système notifie l’utilisateur du type d’étape

Post-conditions :
- Les informations du point d’intérêt sont affichées

Visuel/Écran :





Use cases pour l'acteur Utilisateur enregistré

Se connecter

Titre : Se connecter
Résumé : Le visiteur peut se loguer afin de s'identifier comme utilisateur enregistré.
Acteurs : Utilisateur enregistré, système

Préconditions :

- L'utilisateur a créé un compte.
- L'utilisateur n'est pas encore connecté.

Scénario nominal :

1. L'utilisateur accède à l'application Web ou mobile.
2. L'utilisateur navigue vers la section « Login ».
3. Le système présente un formulaire d'authentification à l'utilisateur.
4. L'utilisateur clique sur l'onglet « Se loguer ».
5. Le visiteur encode son adresse email, son mot de passe et clique « Envoyer »
6. Le système vérifie les informations entrées par l'utilisateur.
7. Le système redirige le visiteur vers l'écran d'accueil.

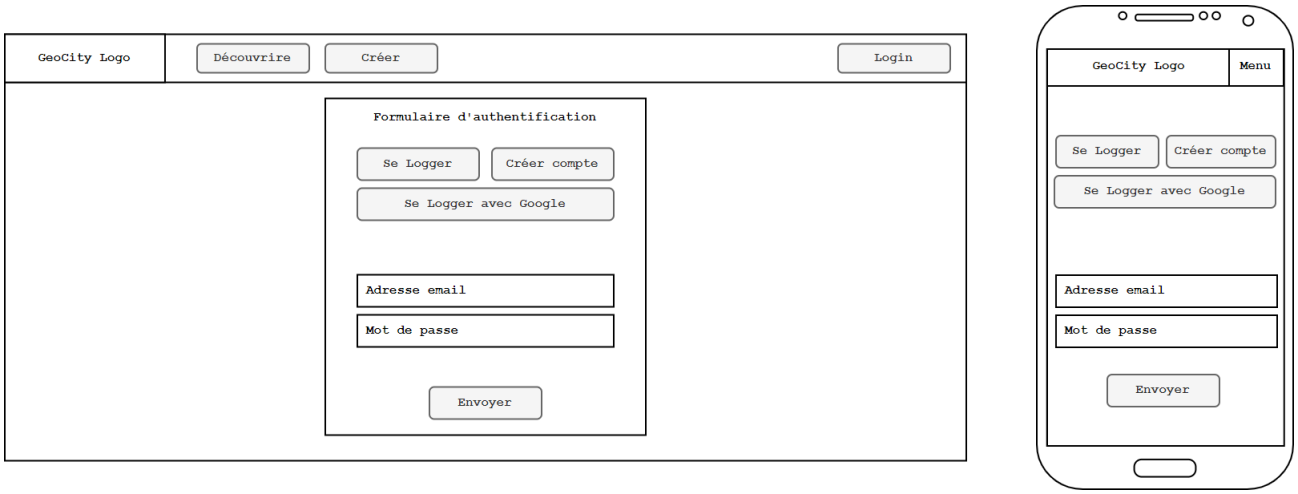
Enchainement d'erreurs :

E1. Le visiteur encode des données erronées.
Démarre au point 5 du scénario nominal.
6. Le système présente au visiteur les erreurs de validation.
7. Le visiteur encode à nouveau ses informations.
Le scénario nominal reprend au point 6.

Post-conditions :

- L'utilisateur est connecté à l'application.

Visuel/Écran :



Noter un citytrip

Titre : Évaluer un citytrip
Résumé : L'utilisateur peut noter un citytrip.
Acteurs : Utilisateur enregistré, système

Préconditions :

- Se connecter
- Afficher les détails d'un citytrip

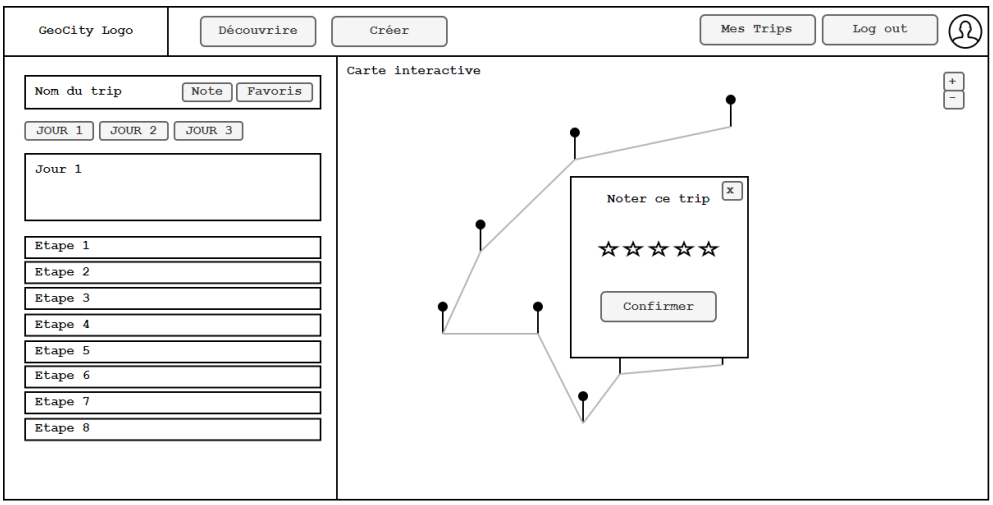
Scénario nominal :

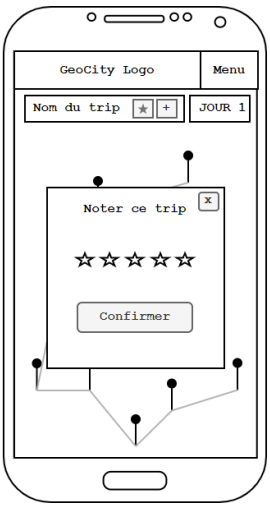
1. L'utilisateur clique sur le bouton « Note ».
2. Le système présente un formulaire à l'utilisateur.
3. L'utilisateur clique sur le nombre d'étoiles qu'il attribue à ce citytrip et confirme.
4. Le système notifie l'utilisateur de la notation du citytrip.

Post-conditions :

- La note globale du citytrip est calculée en fonction de la note attribuée par l'utilisateur.

Visuel/Écran :





Ajouter un citytrip aux favoris

Titre : Ajouter un citytrip aux favoris
Résumé : L'utilisateur peut ajouter un citytrip dans ses favoris.
Acteurs : Utilisateur enregistré, système

Préconditions :

- Se connecter
- Afficher les détails d'un citytrip

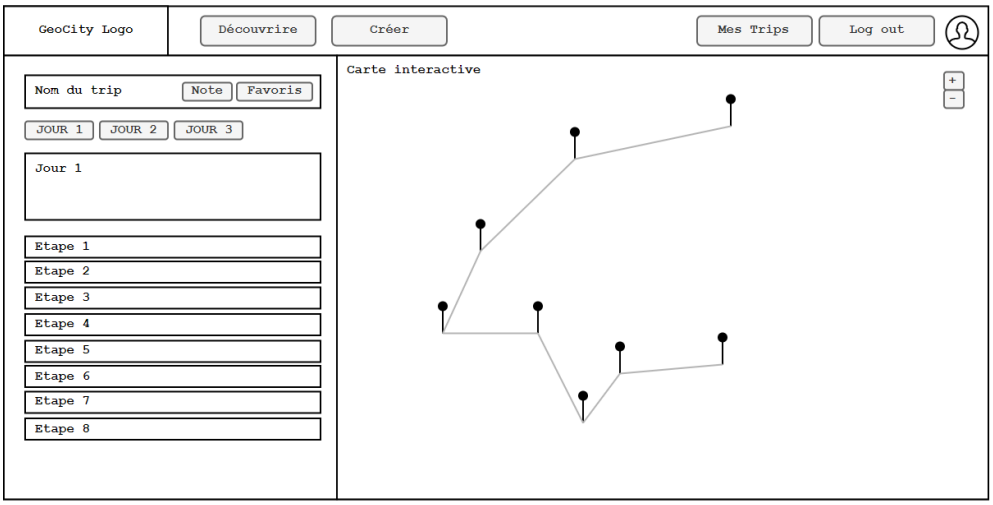
Scénario nominal :

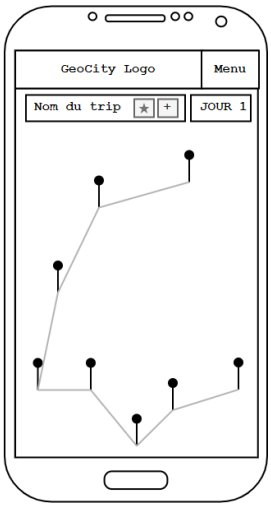
1. L'utilisateur clique sur le bouton « Favoris ».
2. Le système enregistre le citytrip dans les favoris de l'utilisateur.
3. Le système notifie l'utilisateur de l'ajout du citytrip dans ses favoris.

Post-conditions :

- Le citytrip est ajouté aux favoris de l'utilisateur enregistré.

Visuel/Écran :





Afficher ses citytrips

Titre : Afficher ses citytrips
Résumé : L'utilisateur participant peut afficher ses différents citytrips.
Acteurs : Utilisateur enregistré, système

Préconditions :

- L'utilisateur a été ajouté comme participant pour un citytrip.

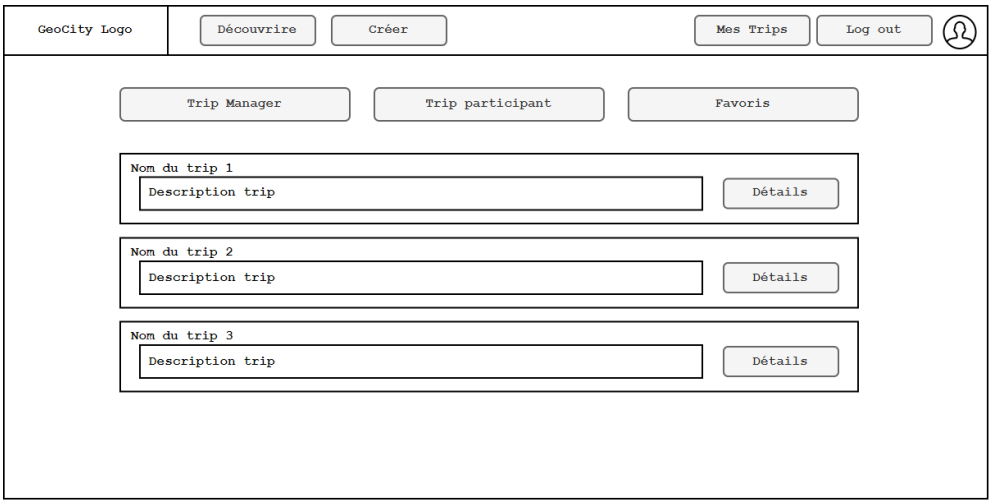
Scénario nominal :

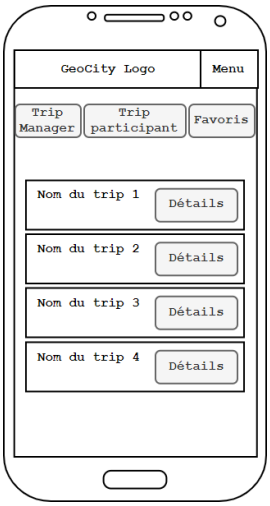
1. L'utilisateur accède à l'application Web ou mobile.
2. L'utilisateur navigue vers la section « Mes citytrips ».
3. Le système récupère les citytrips dans la base de données.
4. Le système redirige l'utilisateur vers l'écran « Mes citytrips ».

Post-conditions :

- Les citytrips sont affichés

Visuel/Écran :





Créer un citytrip

Titre : Créer un citytrip
Résumé : L'utilisateur enregistré peut créer un citytrip.
Acteurs : Utilisateur enregistré, système

Préconditions :
- Se connecter

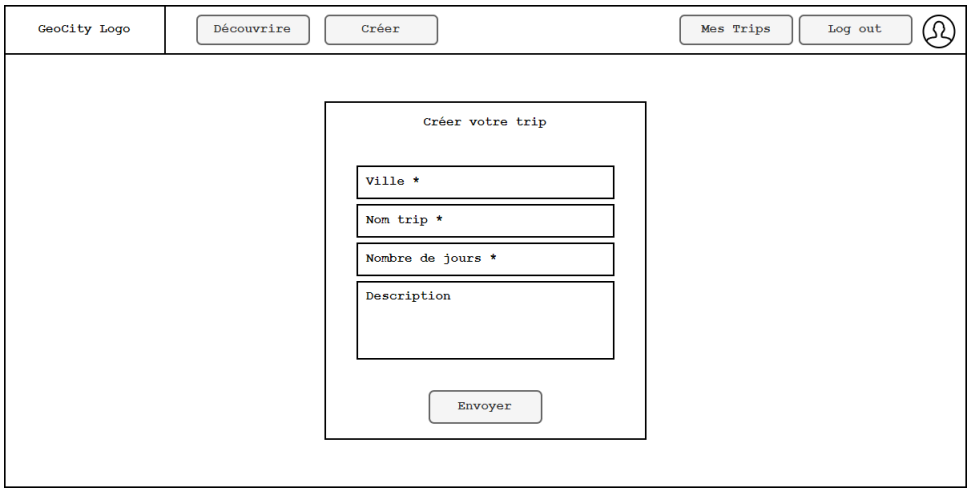
Scénario nominal :
1. L'utilisateur accède à l'application Web ou mobile.
2. L'utilisateur navigue vers la section « créer ».
3. Le système présente un formulaire de création d'un citytrip.
4. L'utilisateur encode les informations du citytrip et clique sur « envoyer ».
5. Le système enregistre le citytrip dans la base de données.
6. Le système redirige l'utilisateur enregistré vers l'écran de configuration de citytrip.

Enchainement d'erreurs :

E1. Le visiteur encode des données erronées.
Démarre au point 4 du scénario nominal.
4. Le système présente au visiteur les erreurs de validation.
5. Le visiteur encode à nouveau ses informations.
Le scénario nominal reprend au point 5.

Post-conditions :
- L'utilisateur enregistré devient utilisateur administrateur pour le citytrip.

Visuel/Écran :



S’inscrire à un citytrip

Titre : S’inscrire à un citytrip
Résumé : L’utilisateur peut s’inscrire à un citytrip comme participant à l’aide d’un lien unique.
Acteurs : Utilisateur enregistré, système

Préconditions :

- L’utilisateur est en possession d’un lien unique d’un citytrip.

Scénario nominal :

1. L’utilisateur entre le lien dans son navigateur.
2. Le système présente un formulaire de connexion.
3. L’utilisateur encode ses informations pour se connecter.
4. Le système authentifie l’utilisateur.
5. Le système enregistre l’utilisateur comme participant au citytrip dans la base de données.
6. Le système redirige l’utilisateur vers l’écran de configuration du citytrip.

Enchainement d’erreurs :

E1. L’utilisateur n’a pas un compte utilisateur.
Démarre au point 2 du scénario nominal.
3. L’utilisateur choisit l’option « créer un compte ».
4. L’utilisateur encode son adresse email et son mot de passe.
5. Le système enregistre les informations dans la base de données.
Le scénario nominal reprend au point 6.

E2. Le citytrip a atteint le nombre maximum de participants.
Démarre au point 4 du scénario nominal.
5. Le système notifie l’utilisateur que le nombre maximum d’utilisateurs participant est atteint.
6. Le système redirige l’utilisateur vers l’écran de détails du citytrip.

Post-conditions :

- L’utilisateur est inscrit comme participant à un citytrip.

Visuel/Écran : voir écran use cases: se connecter, créer un compte, afficher les détails d’un citytrip

Use cases pour l'acteur Utilisateur participant

Afficher les détails d'un citytrip

Titre : Afficher les détails d'un citytrip
Résumé : Le visiteur peut consulter en détail un citytrip et procéder aux configurations.
Acteurs : Utilisateur participant, Système

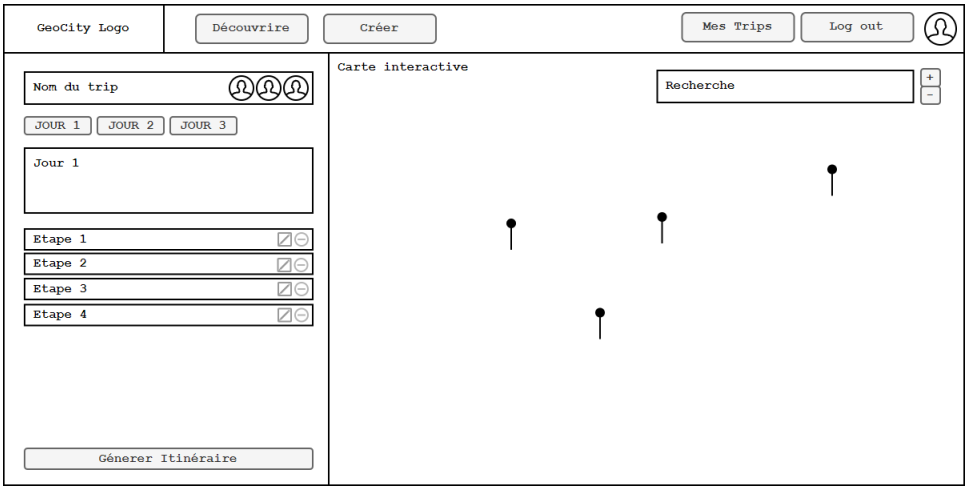
Préconditions :

- Se connecter
- S'inscrire à un citytrip

Scénario nominal :

1. Le participant navigue vers la section « Mes citytrips – citytrip participant ».
2. Le système récupère tout les citytrips auxquels l'utilisateur est inscrit comme participant.
3. L'utilisateur clique sur un des citytrips.
4. Le système redirige le visiteur vers l'écran détails d'un citytrip.

Visuel/Écran :



Ajouter une étape

Titre : Ajouter un point d'intérêt à un citytrip
Résumé : L'utilisateur peut ajouter une étape à un citytrip.
Acteurs : Utilisateur participant, système

Préconditions :

- Se connecter
- Afficher les détails d'un citytrip
- L'utilisateur est un utilisateur participant

Scénario nominal :

1. L'utilisateur clique sur un endroit sur la carte interactive.
2. Le système demande si l'étape ajoutée est un point de passage ou un point d'intérêt.
3. L'utilisateur sélectionne l'option « point d'intérêt ».
4. Le système récupère les informations du point d'intérêt le plus proche du point dessiné.
5. Le système présente un formulaire pour la configuration du point d'intérêt.
6. L'utilisateur encode un prix, une durée et une description.
7. Le système enregistre l'étape dans la base de données.
8. Le système notifie l'utilisateur de l'ajout de l'étape.

Enchainements alternatifs :

A1. L'utilisateur sélectionne l'option « point de passage ».
Démarre au point 2 du scénario nominal.
3. L'utilisateur sélectionne l'option « point de passage ».
Le scénario nominal reprend au point 7.

A2. L'utilisateur effectue une recherche d'un point d'intérêt dans la barre de recherche.
Démarre au point 0 du scénario nominal.
1. L'utilisateur encode le nom d'un point d'intérêt.
2. Le système récupère les points d'intérêt qui correspondent au critère de recherche.
3. L'utilisateur clique sur le lieu qu'il souhaite ajouter.
Le scénario nominal reprend au point 5.

A3. L'utilisateur ajoute un point d'intérêt à partir d'une liste de suggestion.
Démarre au point 0 du scénario nominal.
1. L'utilisateur clique sur le bouton « suggestion ».
2. Le système présente à l'utilisateur un écran avec une liste de suggestion.
3. L'utilisateur clique sur le bouton « + » pour ajouter une étape recommandée.
Le scénario nominal reprend au point 5.

Enchainement d'erreurs :

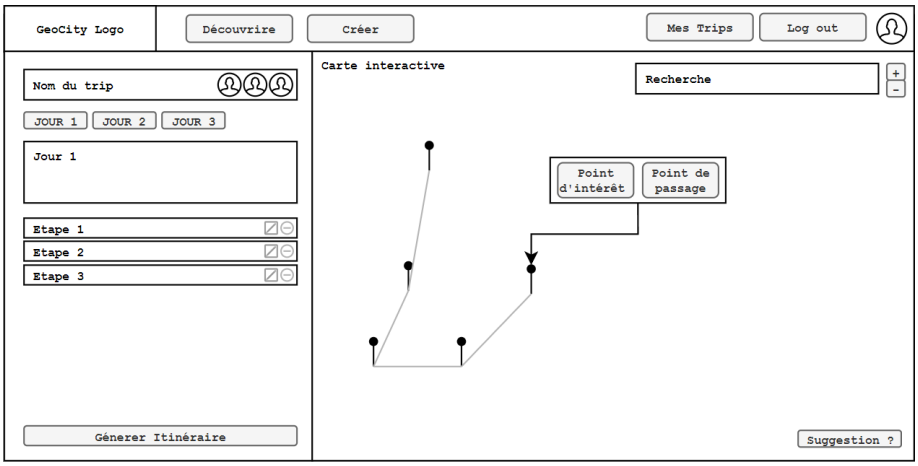
E1. Les informations récupérées par le système ne correspondent pas au point d'intérêt.
Démarre au point 5.
4. L'utilisateur clique sur « quitter ».
Le scénario nominal reprend au point 1.

Post-conditions :

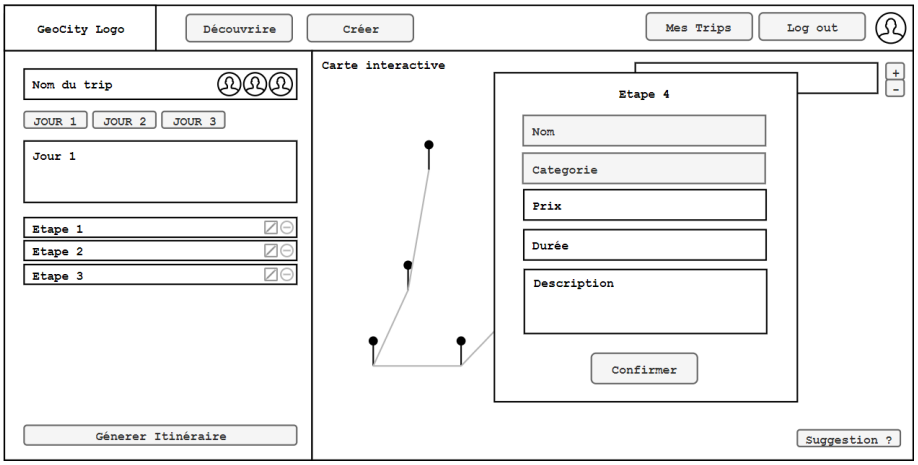
- Une étape est ajoutée à un citytrip

Visuel/Écran :

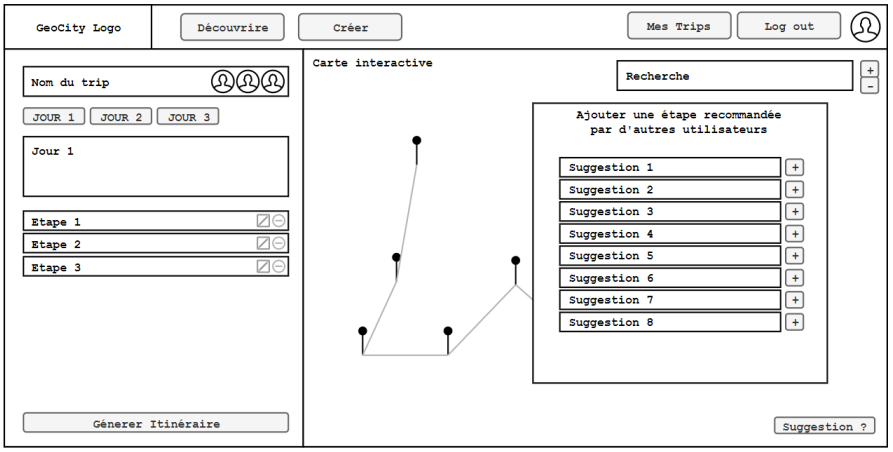
Scénario nominal point 2



Scénario nominal point 5



Scénario alternatif A3



Modifier un point d'intérêt

Préconditions :

- Se connecter
- Afficher les détails d'un citytrip
- L'utilisateur est un utilisateur participant

Scénario nominal :

1. L'utilisateur clique sur une étape dessinée sur la carte interactive.
2. Le système présente un formulaire de modification d'étape.
3. L'utilisateur encode les informations de l'étape et clique sur « confirmer ».

Le système enregistre les informations de l'étape en base de données.

Enchainement d'erreurs :

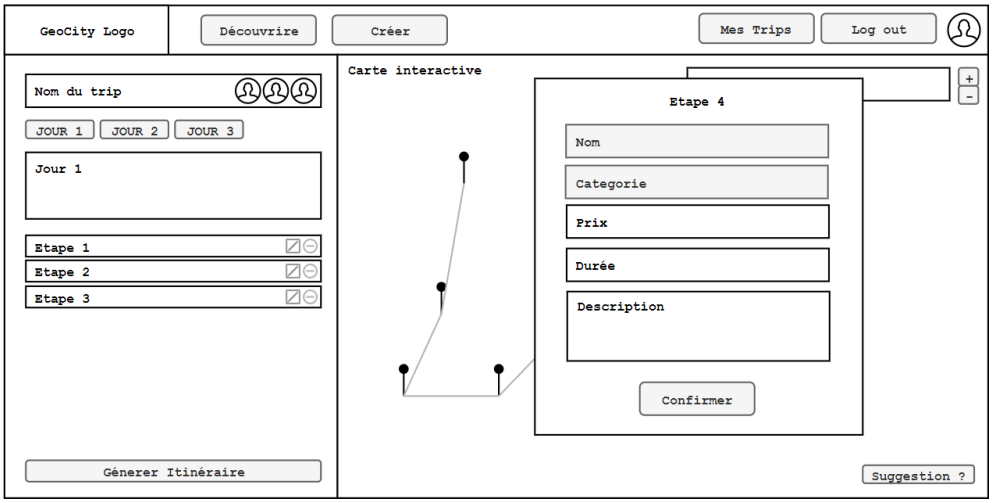
L'utilisateur encode des données erronées.

4. Démarre au point 3 du scénario nominal.
5. Le système présente au visiteur les erreurs de validation.

Le visiteur encode à nouveau ses informations.

Le scénario nominal reprend au point 4.

Visuel/Écran :



Supprimer une étape

Préconditions :
- Se connecter
- S'inscrire à un citytrip
- Afficher les détails d'un citytrip

Scénario nominal :
1. L'utilisateur navigue vers la page de détails du citytrip.
2. L'utilisateur clique sur l'icône de suppression d'une étape.
3. Le système enregistre la suppression de l'étape dans la base de données.
   Le système notifie l'utilisateur de la suppression de l'étape.

Post-conditions :
- L'étape est supprimée

Visuel/Écran : voir écran détail d'un citytrip

Générer un itinéraire

Préconditions :
- Se connecter
- S'inscrire à un citytrip
- Afficher les détails d'un citytrip

Scénario nominal :
1. L'utilisateur navigue vers la page de détails du citytrip.
2. L'utilisateur clique sur le bouton « Générer itinéraire ».
3. Le système récupère la route pour toutes les étapes de l'itinéraire.
4. Le système enregistre la route dans la base de données.
   Le système dessine la route sur la carte interactive.

Post-conditions :
- La route d'un itinéraire est créée

Visuel/Écran : voir écran détail d'un citytrip

Use cases pour l'acteur Utilisateur administrateur

Modifier un citytrip

Titre : Modifier un citytrip
Résumé : L'utilisateur peut modifier les informations d'un citytrip
Acteurs : Utilisateur administrateur, système

Préconditions :

- Se connecter
- Afficher les détails d'un citytrip
- L'utilisateur est un utilisateur administrateur du trip.

Scénario nominal :

1. L'utilisateur clique sur le bouton d'édition d'un citytrip.
2. Le système présente un formulaire de modification du citytrip.
3. L'utilisateur encode les informations du citytrip.
4. Le système enregistre les nouvelles informations dans la base de données.

Le système notifie l'utilisateur de la modification du citytrip.

Enchainements alternatifs : aucun

Enchainement d'erreurs :

E1. Le visiteur encode des données erronées

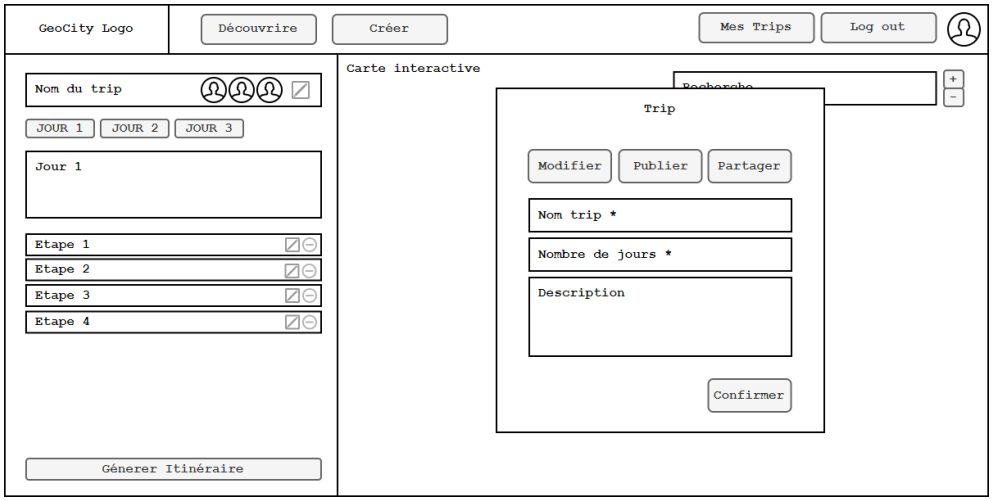
Démarre au point 3 du scénario nominal

4. Le système présente au visiteur les erreurs de validation
5. Le visiteur encode à nouveau ses informations

Le scénario nominal reprend au point 4.

Post-conditions :

Visuel/Écran :



Publier un citytrip

Titre : Publier un citytrip
Résumé : L'utilisateur peut donner son accord pour rendre son citytrip accessible en mode lecture par les autres utilisateurs de l'application
Acteurs : Utilisateur administrateur, système

Préconditions :
- Se connecter
- Afficher les détails d'un citytrip

L'utilisateur est un utilisateur administrateur du trip

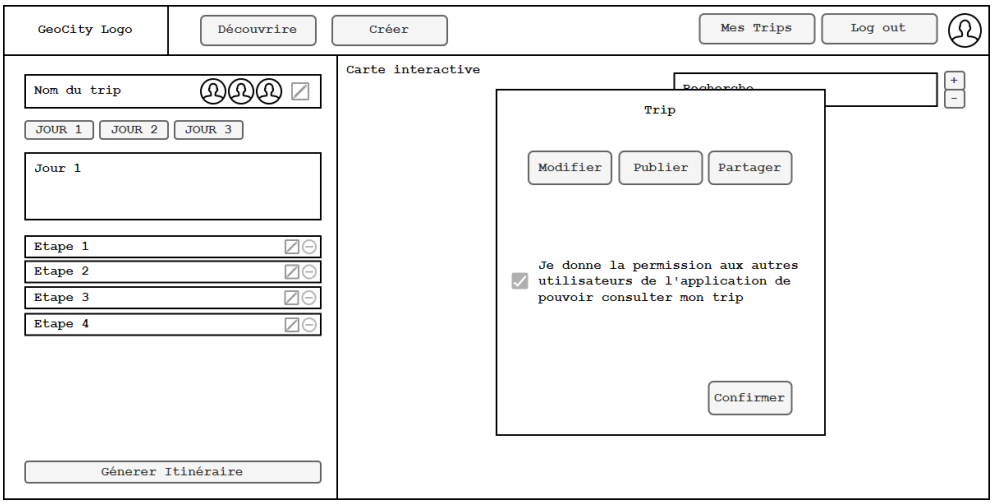
Scénario nominal :
1. L'utilisateur clique sur le bouton d'édition d'un citytrip
2. Le système présente un formulaire de modification du citytrip
3. L'utilisateur cliquer sur l'onglet « Publier »
4. L'utilisateur confirme son intention de publier son citytrip publiquement.
5. Le système enregistre l'information dans la base de données
6. Le système notifie l'utilisateur de la publication de son citytrip.

Enchainements alternatifs : aucun

Enchainement d'erreurs : aucun

Post-conditions :
- Le citytrip est publié

Visuel/Écran :



Partager un citytrip

Titre : Partager un citytrip
Résumé : L'utilisateur partage un citytrip en transmettant un lien unique à un autre utilisateur
Acteurs : Utilisateur administrateur du trip, système

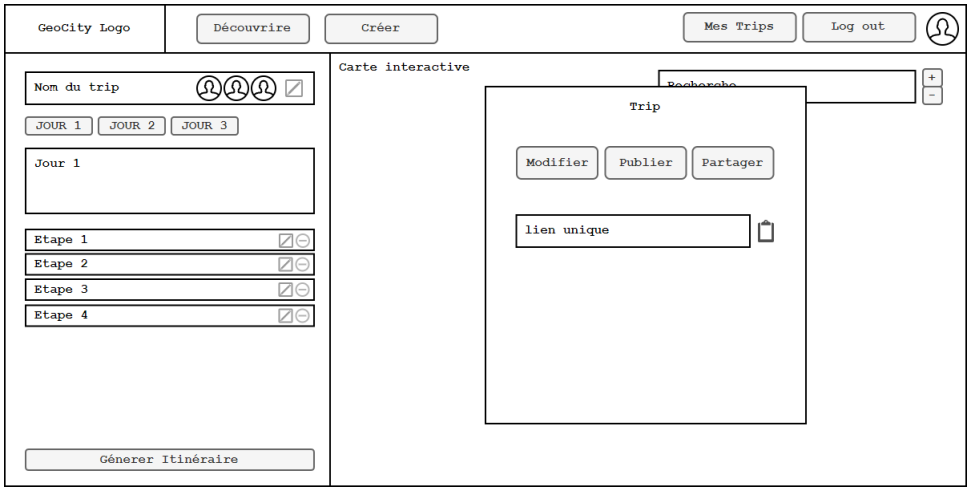
Préconditions :

- Se connecter
- Afficher les détails d'un citytrip
- L'utilisateur est un utilisateur administrateur du trip

Scénario nominal :

1. L'utilisateur clique sur le bouton d'édition d'un citytrip
2. Le système présente un formulaire de modification du citytrip
3. L'utilisateur clique sur l'onglet « Partager »
4. L'utilisateur administrateur transmet le lien unique à un utilisateur enregistré

Visuel/Écran :



Use cases pour l'acteur Administrateur

Afficher les utilisateurs

Titre : Afficher les utilisateurs
Résumé : Un administrateur peut afficher la liste des utilisateurs enregistrés
Acteurs : Administrateur, système

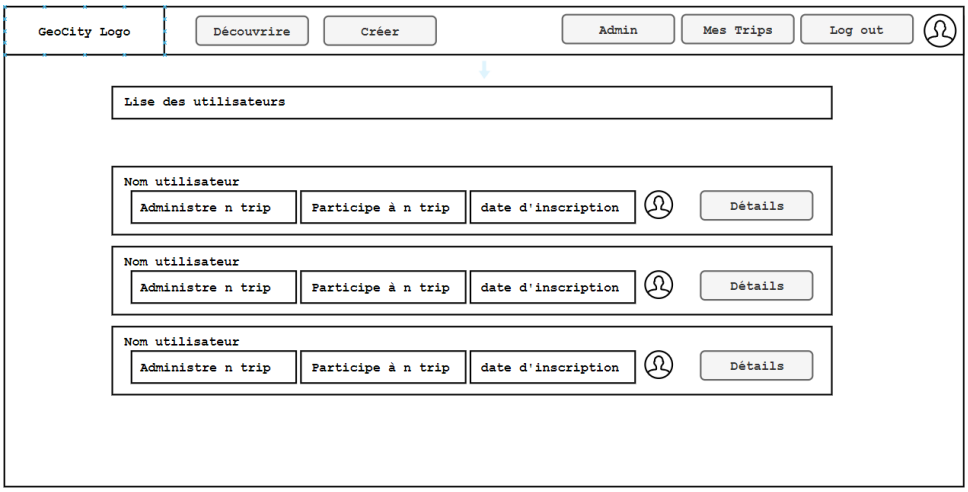
Préconditions :

- Se connecter
- L'utilisateur est un administrateur

Scénario nominal :

1. L'administrateur accède à l'application Web.
2. L'administrateur navigue vers la section « Admin ».
3. L'administrateur navigue vers la section « Utilisateurs ».
4. Le système récupère la liste des utilisateurs enregistrés.

Visuel/Écran :



Afficher les détails d’un utilisateur

Titre : Afficher les détails d’un utilisateur
Acteurs : Un administrateur peut afficher les détails d’un utilisateur.
Acteurs : Administrateur, système

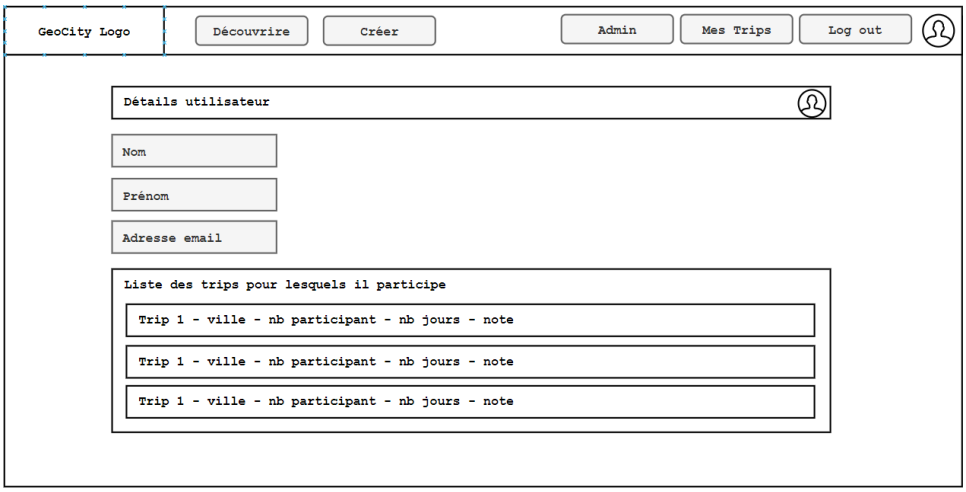
Préconditions :

- Se connecter
- L’utilisateur est un administrateur
- Afficher les utilisateurs

Scénario nominal :

1. L’administrateur clique sur le bouton « détails » d’un utilisateur.
2. Le système récupère les informations de l’utilisateur.
3. Le système récupère la liste des citytrips auxquels l’utilisateur participe.

Visuel/Écran :



Afficher les détails d’une ville

Titre : Afficher les détails d’une ville
Résumé : Un administrateur peut afficher les détails d’une ville.
Acteurs : Administrateur, système

Préconditions :

- Se connecter
- L’utilisateur est un administrateur

Scénario nominal :

1. L’administrateur accède à l’application Web.
2. L’administrateur navigue vers la section « Admin ».
3. L’administrateur navigue vers la section « Villes ».
4. Le système présente un formulaire de recherche.
5. L’utilisateur entre le nom d’une ville et sélectionne la ville.

Le système récupère les informations sur la ville sélectionnée.

Visuel/Écran :

Afficher les suggestions d'une ville

Titre : Afficher les suggestions d'une ville
Résumé : Un administrateur peut afficher les points d'intérêt suggéré pour une ville.
Acteurs : Administrateur, système

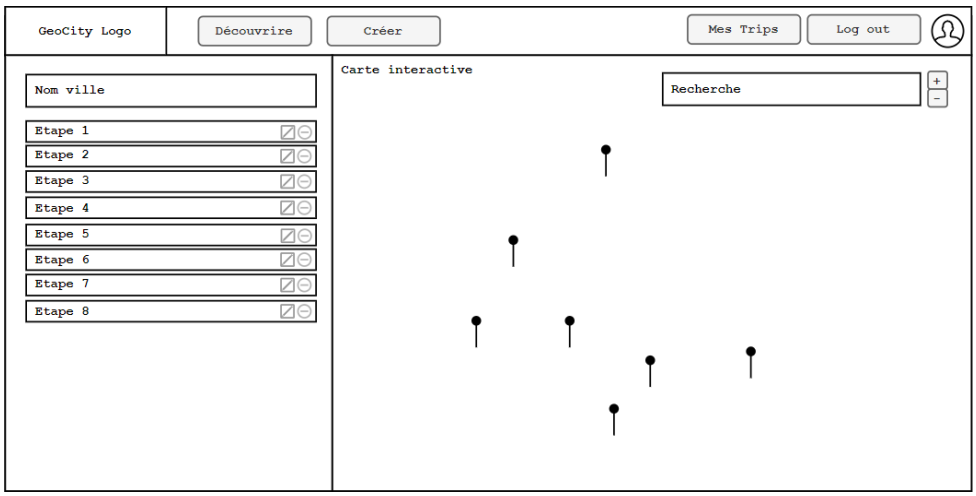
Préconditions :

- Se connecter
- L'utilisateur est un administrateur
- Afficher les détails d'une ville

Scénario nominal :

1. L'utilisateur clique sur le bouton « Suggestions ».
2. Le système récupère la liste des points d'intérêt considéré comme suggestion.
3. Le système dessine les suggestions sur une carte interactive.

Visuel/Écran :



Ajouter une suggestion

Titre : Ajouter une suggestion
Résumé : L'administrateur ajoute une étape comme suggestion
Acteurs : Administrateur, système

Préconditions :

- Se connecter
- L'utilisateur est un administrateur

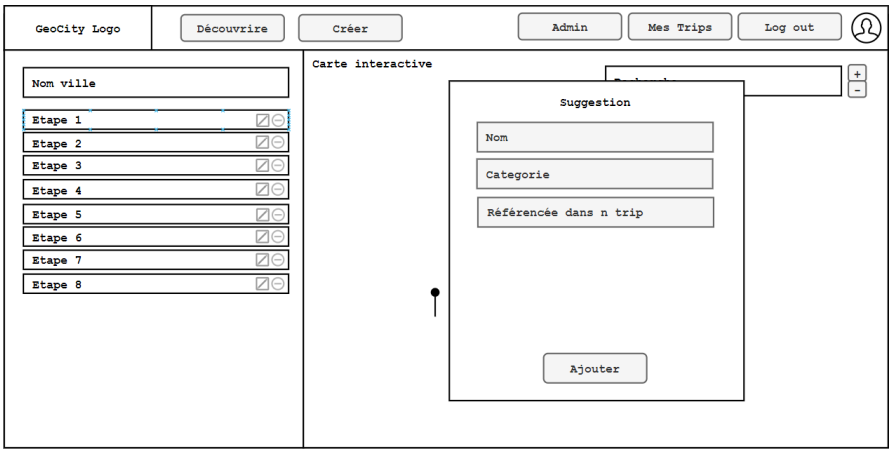
Scénario nominal :

1. L'administrateur accède à la section « Admin » de l'application.
2. L'administrateur navigue vers la section « suggestions ».
3. Le système présente un formulaire permettant de sélectionner une ville.
4. L'administrateur sélectionne une ville.
5. Le système redirige l'administrateur vers le détail des suggestions pour une ville.
6. L'administrateur ajoute une suggestion en cliquant sur la carte.

Le système enregistre la suggestion dans la base de données.

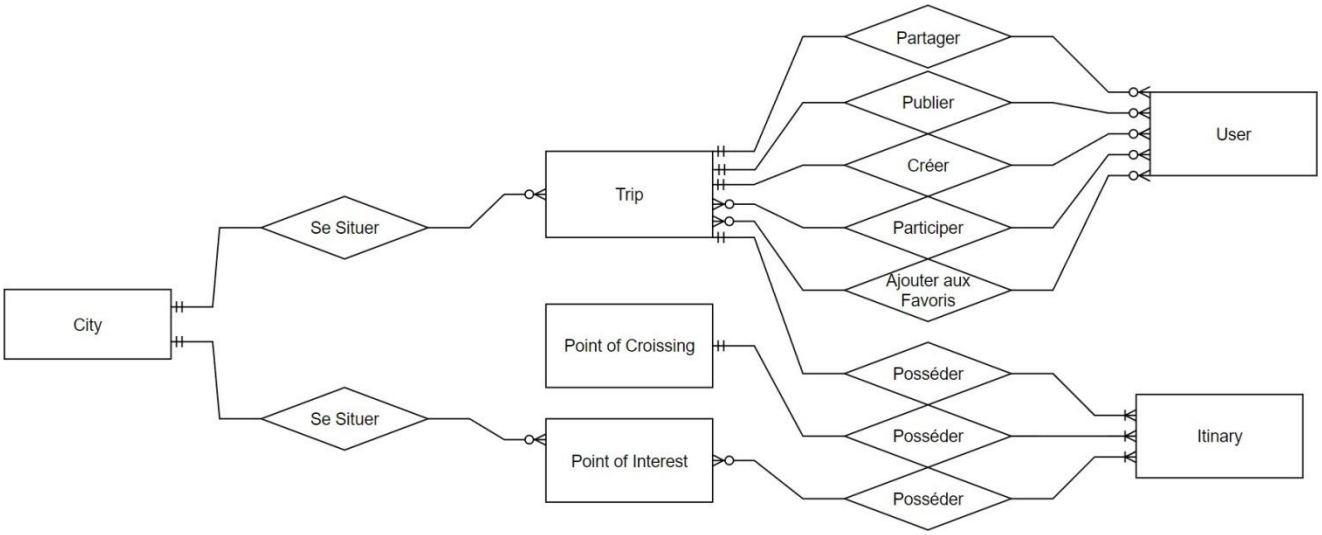
Post-conditions :
Une suggestion/étape est ajoutée dans la base de données.

Visuel/Écran :



4. Base de données

Schéma entité association



Explications

- **trip – itinary [posséder]** : Un citytrip possède un ou plusieurs itinéraires. Cette relation se caractérise par une relation « one-to-many ». Un citytrip peut posséder plusieurs itinéraires, mais un itinéraire est nécessairement lié à un et un seul itinéraire.

- **user – trip [participer]** : un utilisateur peut participer à plusieurs citytrips. Cette relation se caractérise par une relation « many-to-many ». Un utilisateur peut participer à plusieurs citytrips. Un citytrip a nécessairement un ou plusieurs participants. En effet, lorsqu'un utilisateur se connecte pour la première fois, il n'a pas encore u l'occasion de s'inscrire à un citytrip, ou d'en créer un.

- **user – trip [créer]** : un utilisateur peut créer plusieurs citytrips. Cette relation se caractérise par une relation « one-to-many ». Un utilisateur peut créer plusieurs citytrips. Un citytrip est nécessairement créé par un et un seul utilisateur.

- **user – trip [publier]** : un utilisateur peut publier plusieurs citytrips. Cette relation se caractérise par une relation « one-to-many ». Un utilisateur peut publier plusieurs citytrips. Un citytrip est nécessairement publié par un et un seul utilisateur.

- **user – trip [partager]** : un utilisateur peut partager plusieurs citytrips. Cette relation se caractérise par une relation « one-to-many ». Un utilisateur peut partager plusieurs citytrips. Un citytrip est nécessairement partagé par un utilisateur.

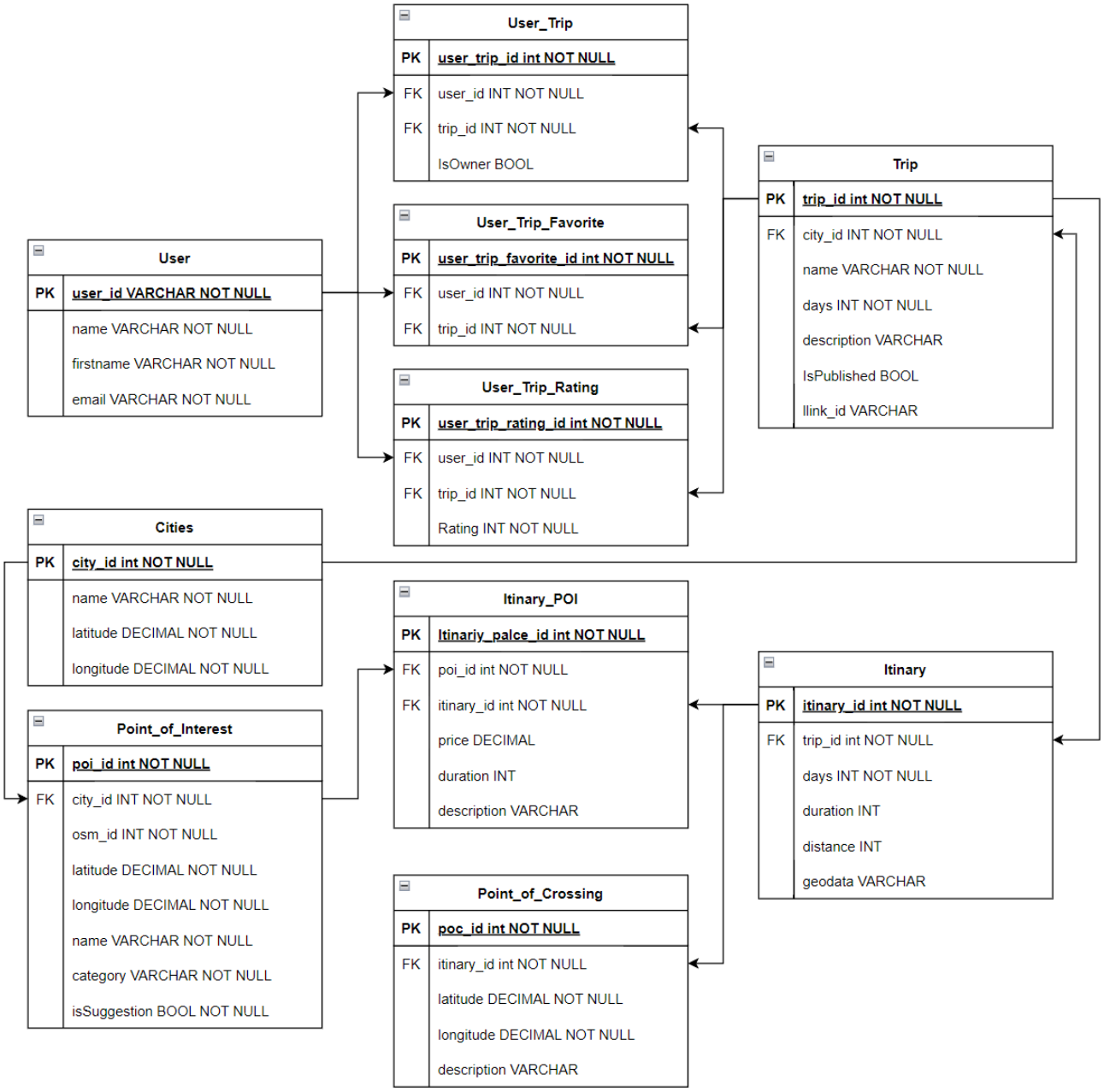
- **user - trip [ajouter aux favoris]** : un utilisateur peut ajouter plusieurs citytrips à ces favoris. Cette relation se caractérise par une relation « many-to-many ». Un citytrip peut être ajouté aux favoris de plusieurs utilisateurs et un utilisateur peut ajouter plusieurs citytrips dans ses favoris.

- **itinary – point of interest [posséder]** : un itinéraire est composé de plusieurs points d'intérêt. Cette relation se caractérise par une relation « many-to-many ». Un itinéraire se compose de plusieurs étapes. Une étape peut composer un ou plusieurs itinéraires. Ainsi, deux utilisateurs différents peuvent utiliser la même étape pour un citytrip différent dans une même ville.

- **itinary – point of crossing [posséder]** : un itinéraire est composé de plusieurs points de passage. Cette relation se caractérise par une relation « one-to-many ». Un itinéraire se compose de plusieurs étapes. Une étape est possédée par un itinéraire. Nous avons décidé de traiter différemment les points de passages des points d'intérêt. Nous justifierions notre décision dans le point suivant.

- **point of interest – city [se situer]** : Un point d'intérêt appartient à une ville. Cette relation se caractérise par une relation « many-to-one ». En effet, plusieurs points d'intérêt se situent nécessairement dans une seule ville.

Schéma relationnel



Passage du schéma entité-association vers relationnel

La relation user – trip

Comme vous pouvez le constater sur le schéma relationnel, nous avons pris la décision de représenter la relation entre un utilisateur et un citytrip à travers trois tables.

- Une table représentant la relation de participation entre un utilisateur et un citytrip.
- Une table représentant la relation entre un utilisateur et ses citytrips favoris
- Une table représentant la relation entre un utilisateur et la note octroyée a un citytrip.

Initialement, nous avions voulu exprimer cette relation à travers une seule table qui comporterait plusieurs attributs représentant la relation entre un utilisateur et un citytrip (ex.: isOwner, isParticipant, isFavorite, Rating). Avec cette approche, la table User_Trip aurait trop de responsabilités différentes.

C'est dans cette optique de séparation de responsabilité que nous avons pris la décision de scinder ces différentes responsabilités en trois différentes tables. Néanmoins, nous avons pris la décision de gérer l'aspect de participation et de création à un citytrip dans une même table puisque ces relations concernent le rôle d'un utilisateur par rapport à un citytrip.

La relation itinary – point of interest – point of crossing

Un point d'intérêt et un point de passage sont clairement dans une relation d'héritage. En effet, ces deux entités peuvent toutes deux être dérivées de l'entité « point » ou « étape », mais elles ont leur particularité propre. Il existe plusieurs manières d'exprimer cette relation d'héritage lorsque nous devons mettre en place nos tables.

Nous avions initialement voulu prendre l'approche de la « Single Table Inheritance ».[1] Cette conception simple consiste à mettre tous les types dans une même table. Ainsi, nous aurions dû créer une table unique ou chaque enregistrement serait soit un point d'intérêt ou un point de passage. Avec cette approche, on retrouve les valeurs communes aux deux entités, mais également un grand nombre de valeurs nulles spécifique pour chaque entité.

Cette approche nous posait problème sur plusieurs plans. Premièrement, les points d'intérêt peuvent obtenir le statut de suggestion. Deuxièmement, les points d'intérêt doivent être nécessairement partagés par les utilisateurs pour rendre compte de leur popularité. Troisièmement, les points d'intérêt doivent avoir une clé étrangère supplémentaire avec leur ville pour permettre à un administrateur d'ajouter un point d'intérêt sans que celui-ci soit préalablement sélectionné par un utilisateur. Quatrièmement, les points d'intérêt comportent plus d'attributs, qui viendraient polluer notre table de valeur nulle.

Nous avons donc opté pour une autre solution, nommée par Karwin dans son ouvrage « SQL antipatterns »[2] « Concrete Table Inheritance ». Avec cette approche, nous créons deux tables distinctes qui partagent un certain nombre de propriétés communes et implémentent leurs propres

[1] Karwin, SQL antipatterns : avoiding the pitfalls of database programming, 2010
[2] idem

propriétés. L'avantage principal de cette approche est d'éviter de stocker des valeurs qui ne s'appliquent pas à un sous-type. Ainsi, notre table « point de passage » n'est pas polluée par des valeurs nulles.

Tables : Règles de structure et de validation

User

Un User représente un utilisateur. Ce dernier est défini par un identifiant, un nom, un prénom et une adresse email. Nous utiliserons dans notre application le service Auth0 pour la création des comptes utilisateurs et l'authentification. C'est Auth0 qui fournira un identifiant unique pour chaque utilisateur.

Règles de structure	
Code	***Description***
RS-001	Un utilisateur à une clé primaire représenté par l'identifiant unique user_id.
RS-002	Un utilisateur à un email unique.
RS-003	Un utilisateur possède obligatoirement un nom, prénom et adresse email.

Trip

La table Trip représente un citytrip. Cette table est définie par un identifiant (clé primaire), un identifiant ville (clé secondaire), un nom, un nombre de jours, une description, une propriété déterminant si le citytrip est publié et un lien unique.

Règles de structure	
Code	***Description***
RS-004	Un citytrip à une clé primaire représenté par l'identifiant unique trip_id.
RS-005	Un citytrip est lié à une ville. Relation représentée par la clé étrangère city_id.
RS-006	Un citytrip possède obligatoirement un nom, un nombre de jours et un lien.
RS-007	Un citytrip n'est par défaut pas publié.
RS-008	Un citytrip à un lien unique.

Règles de validation	
Code	***Description***
RV-001	Un citytrip est composé d'au moins 1 jour.
RV-002	Un citytrip est composé d'au maximum 7 jours.
RV-003	Un citytrip ne peut pas être publié, s'il n'y a pas au moins 3 étapes dans chaque itinéraire.

User_Trip

Un User_Trip représente la relation entre un utilisateur et un citytrip. Cette table est définie par un identifiant (clé primaire), un identifiant utilisateur (clé étrangère), un identifiant citytrip (clé étrangère) et la propriété isOwner. Cette dernière détermine si un utilisateur est un simple participant ou s'il est l'administrateur du citytrip. Nous pourrions envisager dans le futur de définir une série de tables pour gérer les différents rôles entre utilisateurs.

Règles de structure	
Code	***Description***
RS-009	Une liaison entre un utilisateur et un citytrip à une clé primaire représenté par l'identifiant unique user_trip_id.
RS-010	Une liaison utilisateur et citytrip est représenté par l'association entre l'identifiant d'un utilisateur et l'identifiant citytrip.
RS-011	Une liaison entre un utilisateur et un citytrip possède obligatoirement une propriété isOwner.

Règles de validation	
Code	***Description***
RV-004	Un citytrip a un et un seul administrateur
RV-005	Un citytrip ne peut être qu'à 5 participants maximum.

User_Trip_Favorite

Un User_Trip_Favorite représente la relation entre un utilisateur et un citytrip favoris. Cette table est définie par un identifiant (clé primaire), un identifiant utilisateur (clé étrangère), un identifiant citytrip (clé étrangère).

Règles de structure	
Code	***Description***
RS-012	Une liaison entre un utilisateur et un citytrip favori à une clé primaire représenté par l'identifiant unique user_trip_favorite_id.
RS-013	Une liaison utilisateur et citytrip favori est représenté par l'association entre l'identifiant d'un utilisateur et l'identifiant citytrip.

User_Trip_Rating

Un Users_Trip_Rating représente la relation entre un utilisateur et un citytrip noté. Cette table est définie par un identifiant (clé primaire), un identifiant utilisateur (clé étrangère), un identifiant citytrip (clé étrangère) et une note

Règles de structure	
Code	***Description***
RS-014	Une liaison entre un utilisateur et un citytrip noté à une clé primaire représenté par l'identifiant unique user_trip_rating_id.
RS-015	Une liaison utilisateur et citytrip noté est représenté par l'association entre l'identifiant d'un utilisateur et l'identifiant citytrip.

Règles de validation	
Code	***Description***
RV-006	Un utilisateur ne peut pas donner une note de moins de 0
RV-007	Un utilisateur ne peut pas donner une note de plus de 5

RV-008	Un utilisateur ne peut pas donner une note pour un trip auquel il participe

Itinary

La table Itinary représente un itinéraire. Ce dernier est défini par un identifiant (clé primaire), un identifiant citytrip (clé secondaire), un jour, une durée, une distance et une donnée géographique. Un itinéraire est lié à un citytrip. La propriété « donnée géographique » représente la route d'un itinéraire. Le nombre de jours dans la table Citytrips détermine le nombre d'itinéraires qui seront créés pour chaque citytrip. Ainsi, un jour du citytrip correspond à un itinéraire. On utilisera la priorité jour pour déterminer l'ordre chronologique de chaque itinéraire.

Règles de structure	
Code	***Description***
RS-016	Un itinéraire à une clé primaire représenté par l'identifiant unique itinary_id.
RS-017	Un itinéraire est lié à un citytrip. Relation représentée par la clé étrangère trip_id.
RS-018	Un citytrip possède obligatoirement un nombre représentant le jour du trip

Règles de validation	
Code	***Description***
RV-009	Le nombre d'itinéraires correspondant exactement au nombre de jours du trip.

Point_Of_Crossing

La table Point_Of_Croissing représente un point de passage. Un point de passage est défini par un identifiant (clé primaire), un identifiant itinéraire, une latitude, une longitude et une description.

Règles de structure	
Code	***Description***
RS-019	Un point de passage à une clé primaire représenté par l'identifiant poc_id.
RS-020	Un point de passage est lié à un itinéraire. Relation représentée par la clé étrangère itinary_id
RS-021	Un point de passage possède obligatoirement une latitude et une longitude

Règles de validation	
Code	***Description***
RV-010	Un point de passage ne peut pas être placé en dehors de la ville qui le concerne

Point_Of_Interest

La table Point_Of_Interest représente un point d'intérêt. Un point d'intérêt est défini par un identifiant (clé primaire), un identifiant ville, un identifiant OSM, une latitude, une longitude, un nom, une catégorie et la propriété déterminant s'il est une suggestion. Un point d'intérêt est obligatoirement lié à une ville. Cela nous permet d'ajouter des étapes qui ne seraient préalablement pas encore liées à un itinéraire. L'attribut IsSuggestion permet de déterminer si une étape se retrouvera dans les étapes suggérées lors de la configuration d'un citytrip.

Règles de structure	
Code	***Description***
RS-022	Un point d'intérêt à une clé primaire représenté par l'identifiant poi_id.
RS-023	Un point d'intérêt est lié à une ville. Relation représentée par la clé étrangère city_id
RS-024	Un point d'intérêt possède obligatoirement un identifiant OSM, une latitude, une longitude, un nom, une catégorie

Règles de validation	
Code	***Description***
RV-011	Un point d'intérêt ne peut pas être placé en dehors de la ville qui le concerne
RV-012	Un point d'intérêt devient une suggestion lorsqu'il est utilisé au moins 5 fois dans différents citytrips.
RV-013	Seul un administrateur peut directement créer un point d'intérêt qui sera une suggestion
RV-014	Un point d'intérêt ne peut pas être supprimé seulement s'il n'est pas une suggestion et s'il n'est pas utilisé pour un autre citytrip.

Itinary_POI

Un Itinary_POI représente la relation entre un point d'intérêt et un itinéraire. Cette table est définie par un identifiant (clé unique), un identifiant point d'intérêt (clé étrangère), un identifiant itinéraire (clé étrangère), un prix, une durée et une description.

Règles de structure	
Code	***Description***
RS-025	Une liaison entre un point d'intérêt et un itinéraire à une clé primaire représenté par l'identifiant unique itinary_poi_id.
RS-026	Une liaison entre point d'intérêt et itinéraire est représentée par l'association entre l'identifiant d'un point d'intérêt et l'identifiant itinéraire.

Règles de validation	
Code	***Description***
RV-015	Le prix d'un point d'intérêt ne peut pas être moins de 0
RV-016	La durée d'un point d'intérêt ne peut pas être moins de 0

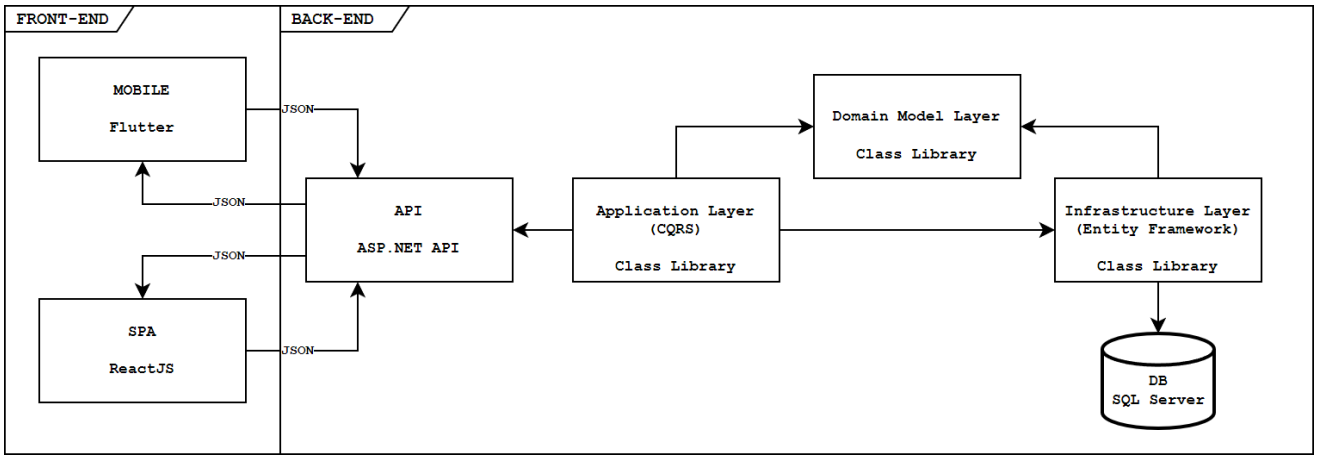
City

La table City représente une ville. Cette dernière est définie par un identifiant (clé primaire), un nom, une latitude et une longitude.

Règles de structure	
Code	***Description***
RS-027	Une ville a une clé primaire représentée par l'identifiant unique city_id.
RS-028	Un citytrip possède obligatoirement un nom, une latitude et une longitude

5. Développement applicatif

Architecture applicative



Côté serveur

Du côté serveur, nous avons pris la décision de développer une API en ASP.NET. Nous utilisons la dernière version du Framework .NET, prénommé .NET 6. Notre code est organisé en différentes couches dérivées du modèle du « *Domain Driven Design* ». Nous faisons également usage du « design pattern » de « *Command Query Responsibility Segregation* » et nous gérons les relations avec notre base de données avec Entity Framework.

Domain Driven Design

Le « Domain Driven Design » est une manière de concevoir son application en se rapprochant le plus possible du domaine business. Ce paradigme de programmation a été initié par Éric Evans, se base sur deux prémisses. Premièrement, l'attention doit être mise sur la logique du domaine. Deuxièmement, les designs complexes du domaine doivent être basés sur un modèle.

L'idée est que la complexité d'un projet ne provient généralement pas des aspects techniques, mais provient du domaine. Il est nécessaire de concevoir une application en suivant la complexité du domaine que l'on tente de circonscrire.[3]

Nous découperons notre projet en 3 différentes couches, « Domain », « Application » et « Infrastructure ». La première couche représente la logique du domaine. Elle inclut toutes les entités spécifiques au domaine. La couche « application » contient toute la logique business. Ce projet implémente le « design pattern » CQRS (Command Query Responsibility Segregation) que l'on présentera ci-après. Cette couche dépend de la couche « domaine ». Enfin, nous retrouvons la couche « Infrastructure », celle-ci contient les services qui permettront d'accéder à des ressources externes telles que notre base de données.

[3]Evans, Domain-driven design, 2004

Command Query Responsibility Segregation

Comme énoncé précédemment, le CQRS est un « design pattern » architectural. Il repose sur le principe de séparation de responsabilité en distinguant les actions de lectures avec les actions d'écriture.

C'est Bertrand Meyer qui introduit pour la première fois la notion de séparation des « commandes » et des « requêtes » dans son ouvrage « *Object Oriented Software Construction* ». Il prescrit que les méthodes des objets doivent être, soit des commandes, soit des requêtes. Ainsi, une requête retourne les données d'un objet sans provoquer de changement à cet objet. Inversement, une commande a pour but de changer l'état d'un objet, mais ne retourne aucune donnée. L'intention est de facilement percevoir et comprendre le but de chaque méthode.

C'est sur ce principe initial que se base le « design pattern » du CQRS. Ce dernier va un peu plus loin et préconise la création de deux objets séparés. Le premier s'occupe des actions de lecture, le deuxième prend en charge les actions d'écriture.

L'implémentation d'un tel paterne permet de rendre l'application plus facilement extensible. En séparant les lectures des écritures, il est désormais possible d'étendre l'un ou l'autre en fonction des besoins. De plus, cette séparation permet de réduire la complexité en ayant d'un côté la logique propre aux lectures et la logique propre aux commandes. Il est également beaucoup plus facile d'ajouter des fonctionnalités, il suffit d'ajouter une nouvelle commande ou requête. Ces dernières ne devraient pas avoir d'impact sur les commandes et requêtes préexistantes[4].

Entity Framework

Afin d'établir une connexion entre la base de données et notre API, nous utilisons Entity Framework en « code first ». Cet ORM (Object Relational Mapper) offre la possibilité de modéliser directement les tables dans la base de données SQL à partir de classes prédéfinies.

Nous utilisons également toutes une série de procédures stockées et triggers. Ces dernières peuvent également être utilisées en combinaison avec les fonctionnalités d'Entity Framework.

[4] Betts & al.,Exploring CQRS and Event Sourcing, 2013

Côté client

Du côté client, plusieurs défis ont dû être résolus. Nos applications client doivent pouvoir inclure des cartes interactives permettant aux utilisateurs de construire leurs itinéraires. Ainsi, le choix d'une API adéquate a été crucial pour assurer le bon déroulement du développement de notre application. Après plusieurs recherches, nous nous sommes naturellement tournés vers OpenStreetMap et le Framework Leaflet pour développer nos cartes interactives. Ces deux API peuvent être utilisés directement à la fois par ReactJS et Flutter.

Analyse et manipulation de données géographiques

Pour mener à bien notre projet, il nous fallait choisir une multitude d'API géographique. Ces dernières pourront nous permettre d'inclure des cartes dans nos applications clients, mais également de créer des itinéraires, ou des retrouver des lieux spécifiques.

Notre choix s'est porté sur OpenStreetMap et les différents services qui ont été développés autour de cette base de données cartographique. Nous aurions pu opter pour l'API de Google ou encore celle de Microsoft, mais OpenStreetMap offre de nombreux avantages. Les couts liés à l'utilisation de l'API de Google ont récemment augmenté, et le projet communautaire entourant OpenStreetMap nous semblait intéressant à explorer.

OpenStreetMap

OpenStreetMap est un projet collaboratif qui vise à établir une base de données cartographique du monde. Ce projet s'inspire du modèle de Wikipedia, ou n'importe quel utilisateur peut contribuer à enrichir la base de données en fournissant des données GPS ou par le biais d'outils spécialisés.[5]

Avec ses 8 millions d'utilisateurs enregistrés, le principal avantage d'OpenStreetMap est sa communauté soucieuse de corriger les contributions inexactes ou erronées, et ainsi d'améliorer la qualité globale de l'ensemble des données.[6] Ces derniers peuvent ajouter de nouvelles informations, telles que des immeubles, point d'intérêt, routes, et autres.

OpenStreetMap ne fournit que des données cartographiques, sans aucun service additionnel. Toutefois, il existe de multiples API développées par la communauté pour permettre aux développeurs d'analyser et manipuler ces données. Ces services permettent entre autres d'inclure des cartes dans des pages Web, ou de montrer des informations supplémentaires en ajoutant des couches supplémentaires.

Pour notre projet, nous aurons besoins
- Un service qui permet d'inclure une carte dans une page Web
- Un service de « Geocoding » qui convertit des données géographiques en adresse lisible, et inversement de pouvoir convertir une adresse en données géographiques.
- Un service de « Routing » qui créé des routes entres différentes étapes.

[5] Teslya, Web mapping service for mobile tourist guide, 2014

[6] Jokar, The emergence and evolution of OpenStreetMap, 2015

Le Geocoding avec Nominatim

Le Geocoding permet de lier des coordonnées géographiques (latitude et longitude) à d'autres données géographiques, telles que des adresses postales ou des codes postaux. Cette opération peut être effectuée dans les deux sens. À partir d'une adresse, nous pouvons avec le Geocoding retrouver des coordonnées ce qui nous permettra de dessiner un point sur une carte. À l'inverse, le « Reverse Geocoding » permet de retrouver une adresse ou un lieu, à partir de coordonnées géographiques.

Nous utiliserons l'API Nominatim pour les opérations de Geocoding. Cette API offre différentes options pour pouvoir répondre à nos besoins. En effet, notre intention est de pouvoir donner la possibilité à nos utilisateurs d'ajouter des étapes à la fois en cliquant sur la carte interactive, mais également en recherchant un lieu précis avec une barre de recherche.

Dans le cas d'une recherche de géocodage inversée, Nominatim à partir de données géographiques retournera l'objet le plus proche du point dessiné. Cet objet provient de la base de données d'OpenStreetMap.

Le Routing avec OSRM

La base de données d'OpenStreetMap comporte des informations sur les routes et chemins et les moyens de locomotion pouvant les emprunter. Il existe de multiples services de routage développé par la communauté, OSRM, GraphHopper, et bien d'autres.[7] Ces services offrent la possibilité de dessiner une route entre plusieurs points. Nous utiliserons dans ce projet l'API d'OSRM qui donne la possibilité de créer des routes et offre de multiples moyens de locomotions. Nous nous contenterons d'utiliser cette API en mode « walking ».

GeoJSON

Ces différents API précités font tous usage du GeoJSON (Geographic Javascript Object Notation). Le GeoJSON est format de données qui structure les informations géo spatiales en utilisant la notion JSON. Ce format permet de décrire plusieurs types d'élément pouvant apparaitre sur une carte, telle qu'un ensemble de points, des lignes, ou autres formes géométriques. GeoJSON est la seule variante géographique de JSON.[8]

Application Web

Nous utiliserons pour ReactJS comme technologie pour créer notre Single Page Application. Nous ferons également usage de ReactLeafet. Cette librairie est une des librairies les plus populaires lorsqu'il s'agit d'inclure des cartes interactives dans une page Web. La première version a été créée en 2010 par Vladimir Agafonkin. Son but était de créer une librairie JavaScript open source pour les utilisateurs voulant créer des applications intégrantes des cartes.[9]

[7] Teslya, Web mapping service for mobile tourist guide, 2014
[8] Horbinski, The use of Leaflet and GeoJSON files for creating the interactive Web map of the preindustrial state of the natural environment, 2022
[9] Edler, The simplicity of modern audiovisual Web cartography, 2019

Leaflet a été conçu pour être facilement utilisable, performant et flexible. Il comprend la majorité des fonctionnalités dont un développeur aurait besoin. Leaflet peut également être étendu à l'aide d'une multitude de plug-ins.[10]

Nous utiliserons les plug-ins « Leaflet Routing Machine » pour gérer les créations des routes pour les itinéraires. Cette librairie s'intègre facilement avec ReactLeaflet. Elle utilise par défaut OSRM, mais il est également possible d'utiliser d'autres services qui permettent de calculer des routes. « Leaflet Control Geocoder » nous permettra d'effectuer les opérations de GeoCoding en utilisant Nominatim.

Application mobile

Notre application mobile sera développée en Flutter. Ce Framework mobile le nouveau langage Dart et permet de créer des applications pouvant être installées à la fois sur iPhone et sur Android. L'équivalent de Leaflet pour flutter se retrouve dans un package appelé flutter_map. Ce dernier tente de récréer les mêmes fonctionnalités de Leaflet.

[10] Wang, The construction of off-line map based on OpenStreetMap and leaflet, 2015

6. Aspect critique

Nous abordons dans cette partie les zones d'amélioration et autres aspects critiques concernant notre projet.

La gestion des rôles doit faire l'objet d'une révision pour améliorer la sécurité de l'application. En effet, nous nous basons sur une seule propriété pour différencier un utilisateur participant d'un utilisateur administrateur. Il serait intéressant de créer de nouvelles tables reprenant les rôles et permissions des différents acteurs. Ceci permettrait d'avoir une approche plus sécuritaire, mais également plus claire des différentes permissions entre utilisateurs. De plus, nous avons seulement considéré le cas, d'un utilisateur administrateur unique pour chaque citytrip. Il serait intéressant de pouvoir avoir plusieurs administrateurs pour un citytrip.

La gestion du partage des citytrips pourrait être revue. Nous avions opté pour la solution du partage d'un citytrip par le biais d'un lien unique, car elle nous paraissait pratique. Néanmoins, à partir du moment où le lien de partage a été transmis, l'inscription des participants au citytrip est hors du contrôle de l'administrateur. Ainsi, n'importe qui ayant obtenu le lien peut potentiellement s'inscrire comme participant. C'est pour cette raison que nous avons décidé de limiter le nombre maximum de participants.

La gestion des publications pourrait être revue. La modification d'un trip publié nous semble devoir faire l'objet de contraintes supplémentaire. Si un utilisateur décide de modifier un trip publié, les utilisateurs qui ont noté, ou, ajouté ce trip dans leurs favoris, devraient en être notifié.

Une attention plus grande devrait également être mise sur les possibles problèmes de transactions dans les cas ou plusieurs participants d'un même trip, tente de modifier ou supprimer une même étape.

Plusieurs fonctionnalités pourraient également être ajoutées. Entre autres, il serait intéressant de pouvoir donner la possibilité aux utilisateurs de sauvegarder les citytrips avec notre application mobile. Ceci permettrait aux utilisateurs d'utiliser notre application sans être connectés à internet. Nous avions également pensé pouvoir donner la possibilité à un utilisateur de copier un trip publié pour ensuite le configurer selon ses propres besoins.

7. Conclusion

Avec ce travail, nous avons voulu proposer une suite d'application permettant à un utilisateur d'organiser des citytrips. Notre but était de créer un projet solide qui puisse répondre de manière efficace aux attentes d'un client potentiel. Une attention particulière a été mise pour rendre l'utilisation de notre application la plus agréable possible en faisant usage de cartes interactives. Ceci nous a permis de découvrir de fascinantes technologies.

L'analyse nous a permis de délimiter les contours de notre projet. Cette étape essentielle de tout projet informatique nous a donné l'opportunité d'identifier et formaliser les différents processus prenant place dans notre application. Nous avons pu ainsi commencer le développement sur de base solide.

Le développement de notre application nous a donné l'occasion d'apprivoiser de nouvelles technologies telles que ReactJS et Flutter, mais également de continuer l'amélioration de nos compétences en .NET. Nous avons également eu la chance de découvrir une nouvelle facette du développement applicatif avec OpenStreetMap et Leaflet.

OpenStreetMap et les services qui en découlent sont des outils puissants et performants qui viennent tenir tête aux géants de l'informatique. Ce fut un plaisir de travailler avec ces technologies open source. Nous recommandons vivement ces technologies aux développeurs désireux de travailler avec des cartes interactives. Nous avons ainsi pu découvrir, le domaine de l'analyse et de la manipulation des données géographiques, qui bien que parfois complexe, est une zone d'étude très intéressante.

Comme nous l'avons mentionné dans le point critique de ce travail, de nombreuses choses peuvent être améliorées. L'aspect sécurité doit définitivement être amélioré en offrant une gestion des rôles plus robuste. Il serait également possible d'ajouter plusieurs fonctionnalités qui offriraient une meilleure expérience aux utilisateurs. Néanmoins, nous considérons ce projet comme une base solide sur laquelle il est possible de construire et implémenter de nouvelles fonctionnalités.

8. Bibliographie

Betts, D., Dominguez, J., Melnik, G., Simonazzi, F., & Subramanian, M. (2013). Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure.

Edler, D., & Vetter, M. (2019). The simplicity of modern audiovisual Web cartography: an example with the open-source javascript library leaflet. js. KN-Journal of Cartography and Geographic Information, 69(1), 51-62.

Evans, E., & Evans, E. J. (2004). *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

Horbiński, T., & Lorek, D. (2022). The use of Leaflet and GeoJSON files for creating the interactive Web map of the preindustrial state of the natural environment. Journal of Spatial Science, 67(1), 61-77.

Jokar Arsanjani, J., Helbich, M., Bakillah, M., & Loos, L. (2015). The emergence and evolution of OpenStreetMap: a cellular automata approach. International Journal of Digital Earth, 8(1), 76-90.

Karwin, B. (2010). SQL antipatterns: avoiding the pitfalls of database programming. Pragmatic Bookshelf.

Teslya, N. (2014, April). Web mapping service for mobile tourist guide. In *Proceedings of 15th Conference of Open Innovations Association FRUCT* (pp. 135-143). IEEE.

Wang, L., Pi, R., Zhou, X., & Zhou, H. (2015, November). The construction of off-line map based on OpenStreetMap and leaflet. In 4th International Conference on Computer, Mechatronics, Control and Electronic Engineering (pp. 1471-1475). Atlantis Press.