
Rapport Final “Here”

Projet informatique individuel

Sommaire

Introduction	3
Contextualisation et objectifs	3
Gestion de projet	4
Organisation et communication	4
Gestion des risques	5
Evolution du planning	6
Informatique	8
Architecture du projet et technologies utilisées	8
Partie 1 : Modèle de données et api web	8
Partie 2 : Application mobile	9
Choix techniques	12
Partie 1 : Modèle de données et api web	12
Partie 2 : Application mobile	13
Test 1 - Managed workflow, react-native-maps & react-native-permissions	14
Test 2 - Managed workflow, react-native-maps & r-n-community/geolocation :	14
Test 3 - Bare workflow, react-native-maps & r-n-community/geolocation :	14
Test 4 - App web, react-native-maps & r-n-community/geolocation :	15
Travail réalisé	17
Partie 1 : Modèle de données et api web	17
Partie 2 : Application mobile	18
Conclusion	23
Bilan fonctionnel et pistes d'évolution	23
Bilan personnel	24

Introduction

Contextualisation et objectifs

Here est un projet d'application mobile à destination des équipes de terrain. Elle s'adresse à tous les utilisateurs qui ont déjà expérimenté des situations où l'organisation et la communication interne devenaient difficiles à gérer et qui se sont vus multiplier les outils et actions afin de s'y retrouver (comme par exemple : partage de localisation sur messenger, google sheet partagé pour le suivi des missions, appels successifs de type : "Allo, t'es où ?", "T'as fini {insérer mission} ?", "Qui s'occupe de {insérer tâche} ?" etc.).

Cette application a donc pour vocation de simplifier l'organisation et la communication interne des équipes de terrain afin de réaliser un réel gain de temps et d'énergie mais surtout d'efficacité et ce, en centralisant toutes les informations nécessaires :

- géolocalisation des membres
- affichage sur google maps
- informations concernant les membres (coordonnées, activité en cours ...)
- informations concernant les missions

Bien que né dans le contexte associatif étudiant, ce projet peut être élargi à de nombreux utilisateurs, tels que des équipes de sécurité ou de secours, ainsi qu'à ceux travaillant dans l'évènementiel de manière plus large.

Gestion de projet

Organisation et communication

Tout au long de ce projet, j'étais encadrée par Jean-Michel Saramito, manager de l'équipe de Système Embarqués chez Thalès.

Deux autres étudiantes étaient encadrées par ce même tuteur et nous nous rencontrions donc un mardi après-midi sur trois, durant le créneau réservé au projet informatique individuel. Plus régulièrement, j'envoyais chaque lundi un compte rendu de mon avancement par mail sous forme de tableau excell (dont le modèle est présenté en annexe à titre indicatif) qui faisait état de lieu :

- des tâches réalisées durant la semaine passée
- des difficultés rencontrées
- de la gestion des risques durant la semaine passée
- des tâches prévues pour la semaine suivante
- des décisions attendues et échéances à court terme
- d'une appréciation globale (bonne, moyenne ou mauvaise) de : la tenue du planning, la gestion des risques, l'avancement technique

Sur conseil de mon tuteur, j'ai utilisé GanttProject afin d'organiser mon planning, de voir les tâches à réaliser, leur avancement ainsi que les jalons importants. Ce logiciel a notamment l'avantage de pouvoir créer des dépendances entre certaines tâches, afin que si une tâche prend plus longtemps que prévu (par exemple), celles qui en héritent et doivent être réalisées après sont automatiquement décalées. Les curseurs d'avancement me permettaient également de voir la progression des tâches prévues. Chaque semaine, je mettais donc à jour mon avancement sur GanttProject et l'exportais en pdf afin de garder une trace de l'évolution de mon planning au cours du temps.

En parallèle, j'utilisais mon application mobile ToDoList afin d'organiser les tâches en plus petites tâches journalières afin de les planifier dans les moments disponibles de mon agenda, d'avoir des notifications de rappel et de suivre leur avancement de manière plus précise.

Gestion des risques

Au début du projet, après avoir identifié les risques possibles, j'ai évalué leurs conséquences et trouvé des solutions envisageables pour ceux qui le nécessitaient. J'ai créé un tableau de suivi des risques et solutions envisagés, et j'ai ensuite réalisé une matrice de risques en évaluant la probabilité et la gravité de chaque risque.

Ces différents tableaux sont disponibles ici : [GoogleSheet - Gestion des risques](#)

Tableau d'identification et de suivi des risques :

	Tâches	Code Risques	Conséquences	Solutions	Probabilité	Gravité	Etat intermédiaire	Etat final	
Risques relatifs aux tâches	Recherches liaisons n-n	RC1	Pas réussir à comprendre le cours GenieLog	Pas de liaison n-n, mauvais modèle	1/ Demander aide à Baptiste Pesquet 2/ Demander aide à Cédric + doc	Faible	Sévère	pas commencé	rencontré et résolu
	Réfléchir modèle + le coder	RB1	Oublier une table, une liaison ou des attributs	Modèle incomplet et/ou non fonctionnel	1/ Anticipation : demander conseil à Cédric en amont 2/ Conserver ce modèle (mais perte de fonctionnalités) 3/ Modifier le modèle (mais grosse perte de temps et d'énergie)	Moyenne	Majeure	en cours de résolution	rencontré et résolu
	Recherches API Google Maps	RA1	Pas réussir à récupérer ancien projet	Pas récupérer l'API Google Maps	1/ Aller voir doc en ligne 2/ Abandonner localisation et passer au plan B	Moyenne	Majeure	pas commencé	rencontré et résolu
	Installer environnement sur machine perso	RM1	Incompatibilités logicielles	Pas réussir à installer sur machine perso	Coder uniquement à l'école (mais perte de temps/flexibilité)	Très faible	Mineure	non rencontré	non rencontré
	Créer et déployer l'application	RA2	Mauvais choix de technologies, packages incompatibles	Pas d'application fonctionnelle	1/ Aller voir doc en ligne 2/ Essayer plusieurs technologies d'application	Moyenne	Sévère	pas commencé	rencontré et résolu
	Apprentissage ReactNative	RC2	Pas comprendre un des cours/TP à rattraper	Prise de retard sur le projet	1/ Demander aide à un élève 2/ Demander aide à Baptiste Pesquet	Moyenne	Moyenne	ok	non rencontré
Risques généraux	/	RM2	Problème de machine perso	Pas de machine perso	Coder uniquement à l'école (mais perte de temps/flexibilité)	Très faible	Mineure	ok	non rencontré
	/	RS1	Rechute de problèmes de santé	Prise de retard sur le projet	Prendre le risque -> moins de fonctionnalités	Moyenne	Moyenne	ok	rencontré et résolu

L'état intermédiaire (avant dernière colonne) correspond au jalon du rendu intermédiaire du 17/03. A ce stade, je n'avais encore rencontré aucun problème critique sur les tâches commencées, et résolu plus ou moins rapidement les autres (voir [Rapport intermédiaire](#)) Cependant à la fin du projet (dernière colonne), il s'avère que le risque RA2 a été rencontré durant la tâche "créer et déployer l'application" et résolu mais beaucoup plus difficilement que les autres, je l'ai donc mis en rouge. En effet, comme je l'expliquerai dans la suite de ce rapport, de nombreuses incompatibilités technologiques ont considérablement retardé et compliqué le projet.

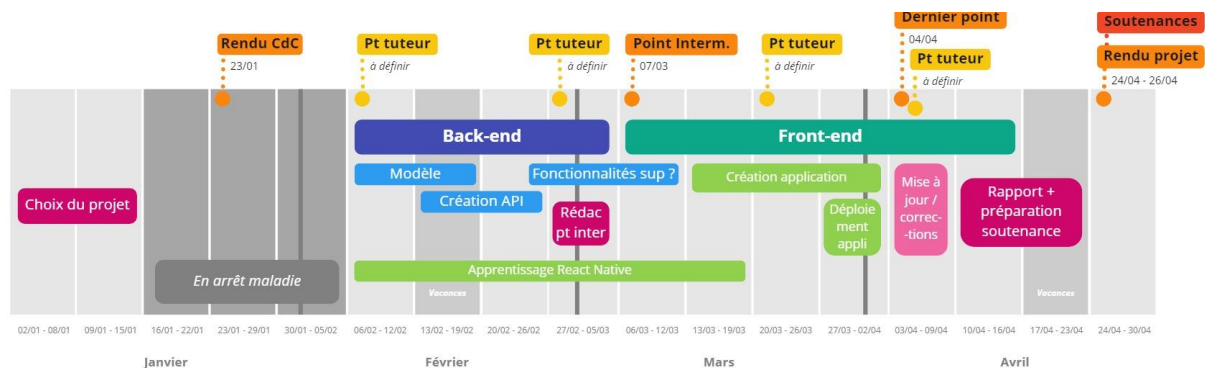
Matrice de criticité des risques :

Matrice des risques		Gravité				
		Négligeable	Mineure	Moyenne	Majeure	Sévère
Probabilité	Forte					
	Moyenne			RC2 RS1	RB1 RA1	RA2
	Faible					RC1
	Très faible		RM1 RM2			

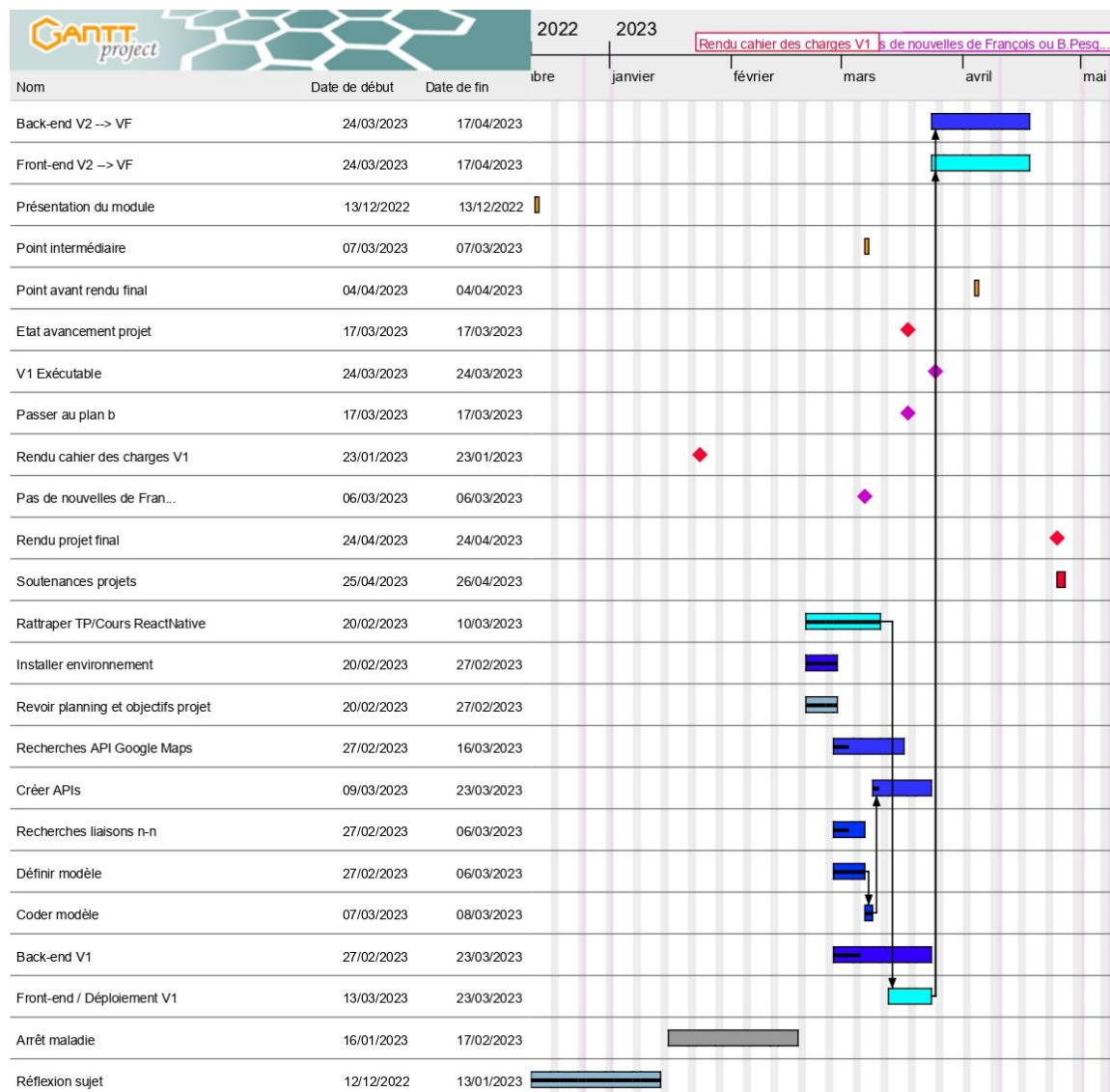
On voit bien sur cette matrice que le risque RA2 précédemment cité avait bien été identifié comme le plus critique (et à raison !).

Evolution du planning

Planning initial (réalisé avec Miro le 23/01) :



Planning intermédiaire (réalisé avec GanttProject le 17/03) :

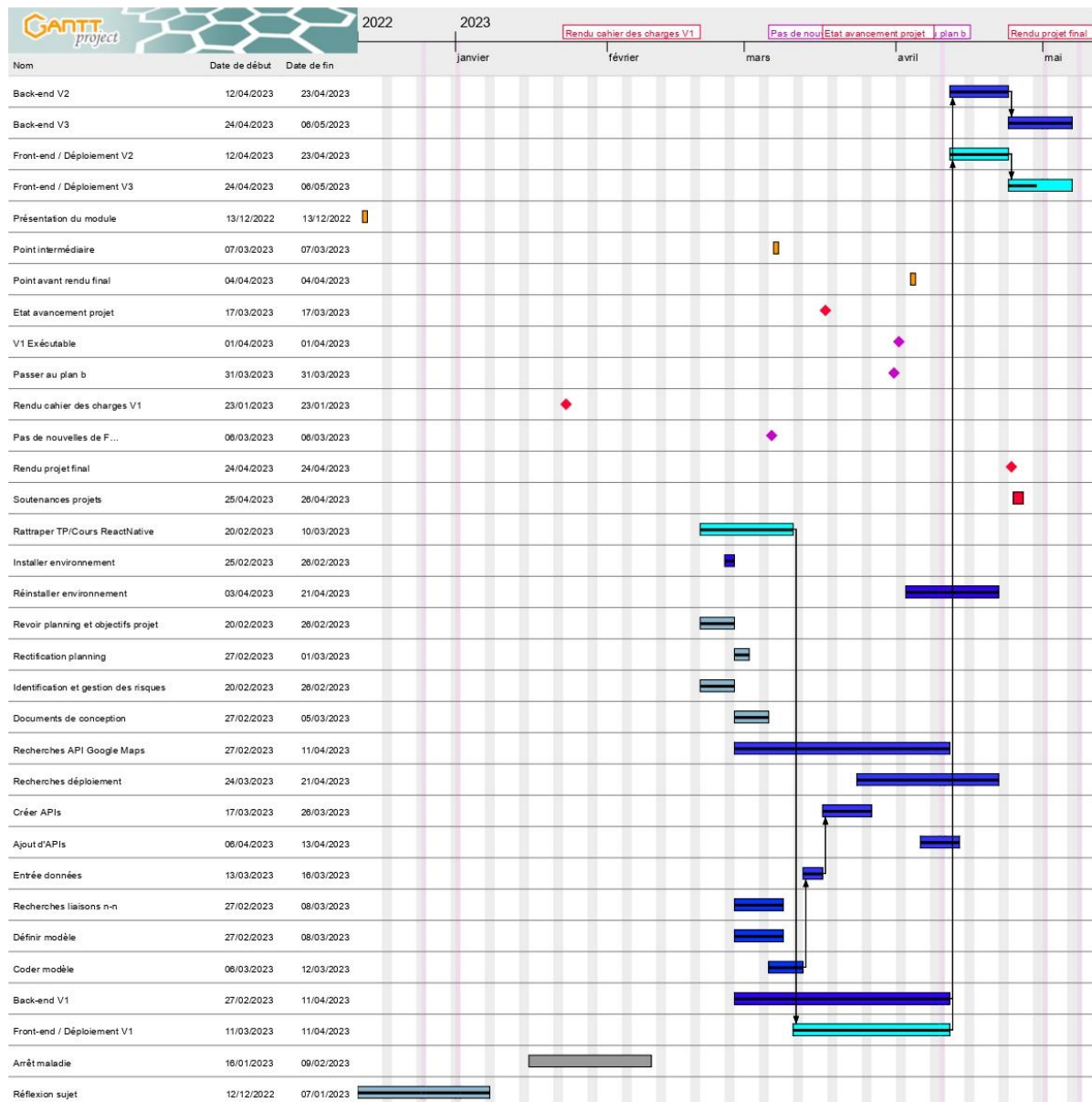


Grâce aux retours et conseils de mon tuteur, j'ai pu apporter plusieurs modifications à mon planning, que j'ai finalement refait avec GanttProject. D'autre part, des problèmes de santé

et un arrêt maladie plus long que prévu (un mois et demi) m'ont contraint à commencer mon projet plus tard et à reporter plusieurs tâches dans le temps. J'ai également pris en compte les risques précédemment cités et les étapes du projet identifiées grâce à mon document de conception : [Conception PII V1](#) .

Ce planning correspond à l'état de mon avancement au 17 mars, lors du point intermédiaire, dont le rapport est ici : [Rapport état avancement V1](#) .

Planning final :



Finalement j'ai pu tenir un planning proche de celui estimé lors du point intermédiaire. J'ai cependant dû faire des choix et restreindre les fonctionnalités prévues en passant au plan b et en me concentrant sur les fonctionnalités principales (voir la section "objectifs" du [cahier des charges](#)).

Informatique

Architecture du projet et technologies utilisées

Afin de réaliser ce projet, j'ai décidé d'utiliser en priorité les technologies étudiées durant mon cursus à l'ENSC, afin de consolider mes connaissances et surtout de développer mes compétences en développement mobile. En effet, les téléphones sont actuellement au cœur de nos vies et les applications mobiles sont à mon sens un domaine de développement particulièrement prometteur qui m'attire beaucoup, notamment au regard de l'expérience utilisateur.

Ainsi j'ai réalisé :

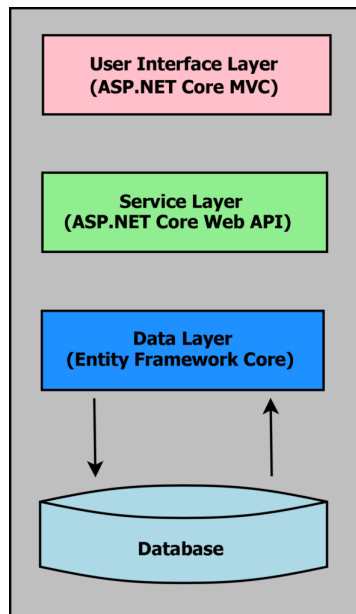
1. un premier projet qui constitue une partie du **back-end**, avec notamment le modèle de données : classes métiers, données brutes et génération de la BDD, ainsi que les api web nécessaires à mon projet
2. un second projet d'application mobile (qui fait appel au modèle et aux api du premier projet) et comporte l'autre partie du **back-end** (avec notamment l'utilisation de l'api Google Maps) ainsi que le **front-end**

Partie 1 : Modèle de données et api web

Le premier projet qui constitue une grande partie du back-end (à savoir le modèle et les api web de mon projet) a été réalisé en utilisant les technologies vues au semestre 7 en Génie Logiciel qui sont :

- l'O/RM Entity Framework Core afin de faire le lien automatiquement avec la base de données gérée par le SGBDR SQLite à partir des modèles et objets codés
- le framework ASP.NET Core MVC afin de générer des API web qui permettent d'interagir avec le système au moyen des protocoles HTTP

Schéma du fonctionnement de Entity Framework Core et ASP.NET Core :



→ Database :

Ici la base de données est gérée par SQLite comme expliqué précédemment.

→ User Interface Layer (ASP.NET Core MVC) :

Un MVC est un Model-View-Controller. Dans ce projet je n'ai finalement utilisé que les api controller (Web API) et pas de controllers base ni de vues (Views) puisque les appels des api et le front-end sont développés avec React Native (voir explications du paragraphe suivant).

Le bloc du dessus devrait donc être :

“User Interface Layer (React Native Expo-cli)”

Partie 2 : Application mobile

Enfin, le second projet qui constitue l'application mobile à proprement parler et développe l'autre partie du back-end ainsi que le front-end a été réalisé en utilisant les technologies étudiées au cours du semestre actuel en Développement Mobile mais surtout grâce à des recherches supplémentaires effectuées. Il utilise donc

- le framework multi-plateforme React Native avec :
 - le langage JavaScript : pour l'appel et l'utilisation des api déployées sur le web, l'implémentation de différents packages et de l'api google maps (générée au préalable via Google Cloud for Developers), ainsi que la création et gestion des écrans et de la navigation
 - un dérivé du langage css : pour le graphisme de l'application
 - le framework Expo-cli qui est un framework “managé” afin de créer et développer l'application plus facilement
 - divers packages et librairies, et notamment : expo-location, react-native-maps, ainsi que l'api Google maps (au moyen d'une clé d'api créée sur un compte Google Cloud)
- L'intérêt de ces packages sera détaillé plus loin dans la section “Choix techniques”.*

Schéma du fonctionnement d'une application React Native :

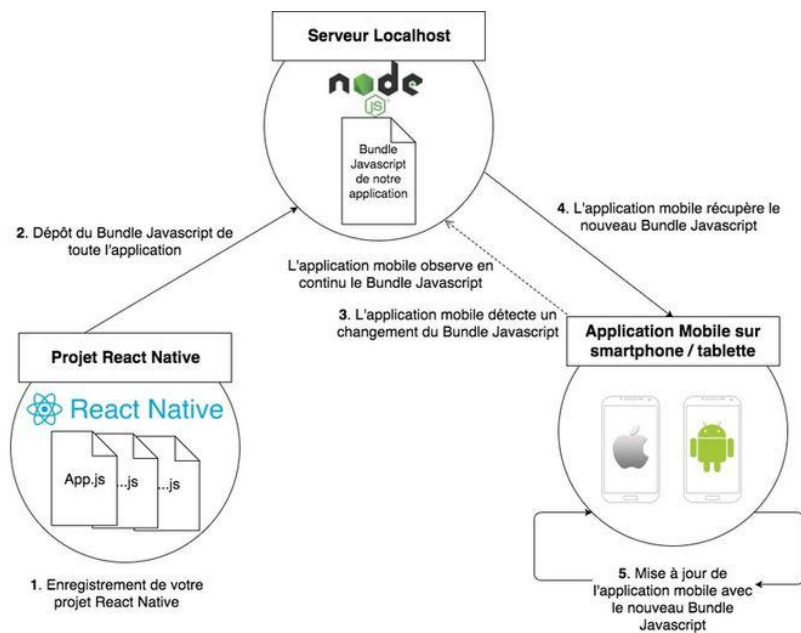


Schéma de fonctionnement du déploiement avec Expo :

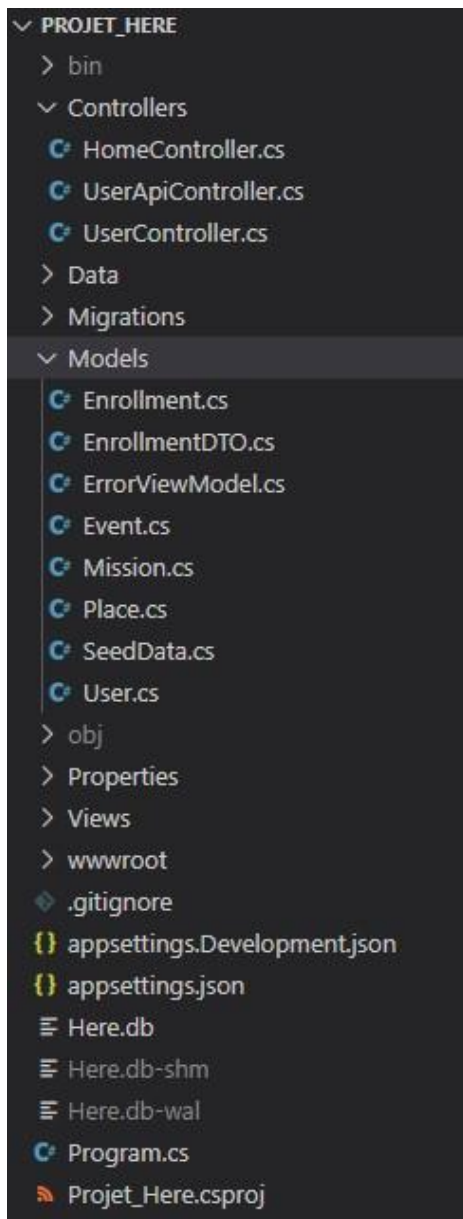


Pour l'utilisateur, il lui suffit de télécharger l'application Expo Go sur son smartphone et de récupérer l'application qui tourne en local sur un ordinateur par scan d'un QR code en wifi, ou bien par tunnelisation (afin de pouvoir se déplacer dans la ville).

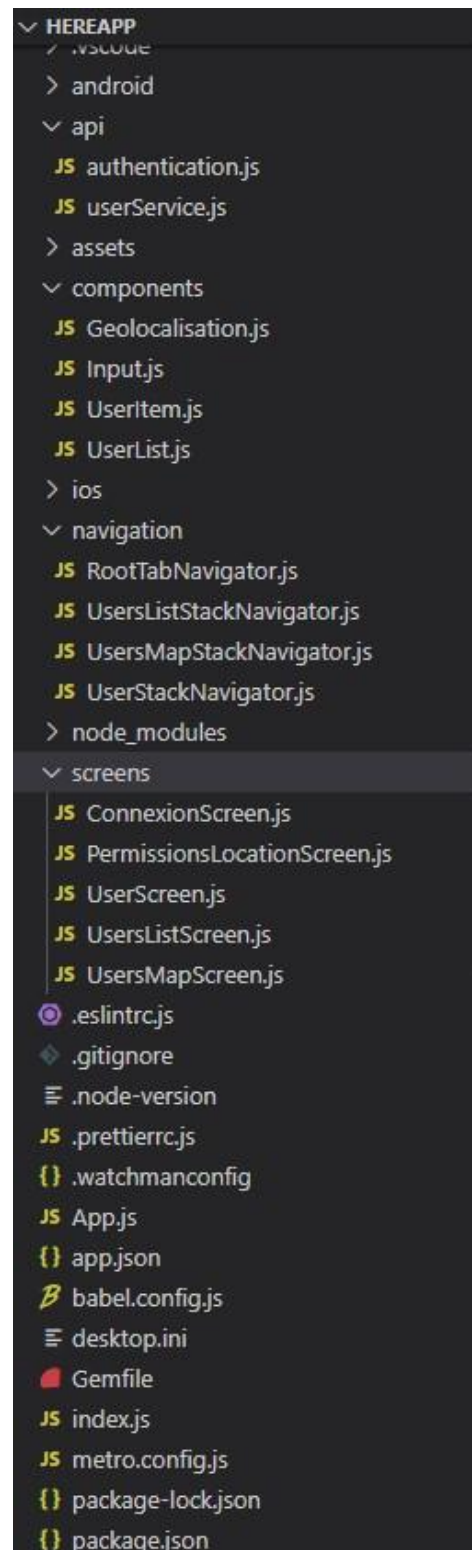
Finalement, j'ai donc 2 projets réalisés séparément et localisés sur 2 repositories différents sur GitHub, mais mis en liaison grâce aux api web que j'ai déployé sur internet grâce à Azure.

Architectures respectives des deux projets :

Modèle de données et apis web



Application mobile



Choix techniques

Partie 1 : Modèle de données et api web

Pour la réalisation du modèle, de la base de données et des api de cette première partie, j'ai dû faire plusieurs choix.

Dans un premier temps, j'avais besoin d'une liaison plusieurs-plusieurs entre ma table mission et ma table user afin qu'un utilisateur puisse participer à plusieurs missions, mais aussi qu'une mission puisse être réalisée par plusieurs utilisateurs.

Ce format de liaison m'a particulièrement posé problème puisque je souhaitais que l'utilisateur puisse pour chaque mission :

- l'accepter (une fois qu'elle lui est assignée)
- la commencer
- la déclarer terminée
- demander de l'aide.

Or, Entity Framework Core (EFCore) mappe automatiquement la base de données en fonction du modèle et crée donc une table *UserMission* qui traduit cette liaison plusieurs-plusieurs entre mes deux tables en récupérant l'identifiant du user et celui de la mission. Bien qu'étant un réel gain de temps et d'énergie pour le mapping de mes autres tables (puisque je n'ai pas besoin de requêtage SQL), ce framework me posait problème puisque je ne pouvais modifier la table *UserMission* et donc encore moins lui ajouter les champs : "accepted", "started", "ended" et "needHelp".

J'ai donc fait le choix de modifier mon modèle en ajoutant une classe métier *Enrollment* qui récupère les identifiants du user et de la mission souhaités et qui contient également les propriétés "accepted", "started", "ended" et "needHelp".

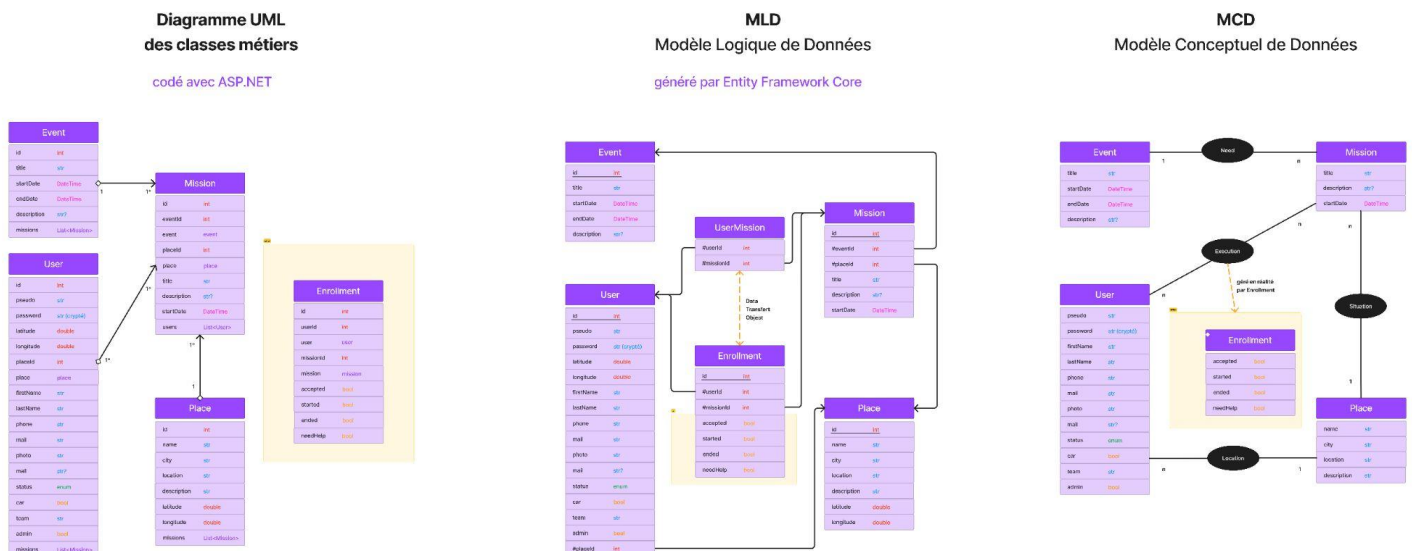
Cette classe est donc mappée sous la forme d'une table *Enrollment* grâce à EFCore. Elle semble faire doublon avec la table *UserMission* mais est en réalité très utile pour mes api.

En effet elle me permet de créer une api de modification (avec PUT) qui agira à la fois sur :

- la table *User* (en ajoutant la mission associée à la liste de missions)
- la table *Mission* (en ajoutant l'utilisateur associé à la liste d'utilisateurs)
- la table *UserMission* (qui récupère les id de user et mission en clés étrangères)
- la table *Enrollment* (qui modifie les valeurs des booléens : "accepted", "started" ...)

Cette api permet finalement à l'utilisateur de modifier le statut de la mission en l'acceptant, la commençant etc. Elle doit être appelée par un formulaire dans l'application (en permettant notamment à l'utilisateur de cocher les cases souhaitées suivant ce qu'il fait de la mission qui lui est assignée).

Schémas du modèle de données réalisé :



Partie 2 : Application mobile

Pour la réalisation de mon application mobile, je souhaitais faire fonctionner les trois fonctionnalités suivantes qui représentent le coeur de mon projet :

- géolocalisation de l'utilisateur
- affichage de la carte et de la position
- déploiement de l'application sur plusieurs smartphones simultanément

J'ai cependant rencontré de nombreuses difficultés techniques liées à des incompatibilités et j'ai donc essayé de créer mon application avec différentes technologies.

Au total, j'ai utilisé 5 technologies différentes pour créer mon application et ai donc développé 5 applications différentes. Ces différents essais successifs m'ont fait réaliser un gros travail informatique concernant l'apprentissage et la prise en main de nouvelles

technologies, l'installation / désinstallation / modification d'environnement et surtout la lecture de documentation.

Test 1 - Managed workflow, react-native-maps & react-native-permissions

J'ai tout d'abord souhaité utiliser le framework "managé" Expo qui permet de créer et de déployer une application plus facilement et avec lequel j'avais travaillé en cours de développement mobile. J'ai cependant rencontré un problème majeur : la géolocalisation du package react-native-maps dépendait d'un autre package de permissions : react-native-permissions (afin que l'utilisateur puisse choisir ou non de laisser l'application accéder à certains de ses composants natifs : à savoir ici la localisation de l'appareil) et ce package s'est avéré incompatible avec mon environnement.

C'est un package réalisé bénévolement par un membre de la communauté Github et vraisemblablement son code n'était pas assez robuste ou bien tout simplement obsolète (il n'avait effectivement pas été mis à jour depuis 6 ans).

Test 2 - Managed workflow, react-native-maps & r-n-community/geolocation :

J'ai ensuite essayé de développer une nouvelle application, toujours en utilisant Expo mais cette fois-ci en utilisant le package react-native-community/geolocation et en n'utilisant react-native-maps que pour l'affichage de la carte et des marqueurs de position. Malheureusement ce package de géolocalisation ne fonctionnait pas avec un framework managé (donc Expo).

Test 3 - Bare workflow, react-native-maps & r-n-community/geolocation :

J'ai donc créé une app react-native avec un framework "bare" (non managé). Cette application me permettait bien d'utiliser le package geolocation ainsi que celui de maps mais le déploiement était beaucoup plus compliqué puisqu'il n'y avait pas de QR code généré et que je ne pouvais tester mon application qu'avec un appareil : mon smartphone en mode dev et debug USB ou bien mon simulateur android de Android Studio. Le seul moyen de déployer mon application sur plusieurs machines (ce qui constitue le cœur de mon projet puisque l'application est à destination d'équipes et non d'utilisateur isolé) était de la déployer sur le Google Play (ou

AppStore pour ios). Or cette solution ne me permettait pas d'implémenter les build successifs et de mettre à jour mon application au fur et à mesure de l'ajout de nouvelles fonctionnalités (en effet le déploiement peut prendre jusqu'à 10 jours et est donc à destination de versions terminées et fonctionnelles).

J'ai en parallèle découvert CodePush qui est censé me permettre de réaliser ces déploiements successifs mais je n'ai pas compris comment l'utiliser.

Test 4 - App web, react-native-maps & r-n-community/geolocation :

J'ai finalement voulu créer une application web (en utilisant react-native-web ainsi que react-scripts et react-dom) afin de contourner le problème de tests sur plusieurs smartphones.

Cette version me permettait bien de déployer et tester l'application sur plusieurs smartphones en allant sur l'url généré à travers le navigateur du téléphone, mais elle ne permettait pas d'utiliser correctement react-native-maps et l'api Google maps qui dépendent des composants natifs de l'appareil.

Test 5 Solution finale - Managed workflow, react-native-maps & expo-location :




Après de nombreux essais infructueux et surtout un gros travail de documentation, j'ai récemment trouvé une solution. Il s'avère que le workflow Expo contient lui-même un package de localisation que je n'avais pas trouvé au préalable. J'ai donc créé une application react-native en workflow "bare" afin de faire fonctionner les fonctionnalités de géolocalisation et affichage sur maps. Et j'ai ensuite converti cette application vers Expo en workflow "managé" et réinstallé le package expo-location afin de remplacer celui de react-native-community/geolocation.

Contre toute attente (en tout cas pour moi), ce travail concernant le choix des technologies, la lecture de documentation et l'installation d'environnement a représenté environ 95% de mon temps de travail contre seulement 5% de code "pur". J'ai cependant acquis de nombreuses compétences informatiques de cette manière ainsi que des connaissances concernant le fonctionnement de nombreuses librairies et packages.

J'ai donc réalisé une brève analyse comparative des différentes technologies que j'ai pu utiliser afin de rendre compte des fonctionnalités que je suis parvenue à développer et des connaissances que ces technologies m'ont apportées.

Tableau comparatif des fonctionnalités implémentées suivant les technologies utilisées :

NB : ce tableau représente l'état actuel de mes propres apprentissages et compétences acquises, il n'est pas représentatif de ce qui peut être réalisé par des professionnels

Objectifs principaux de l'application					
Fonctionnalité	Accès à la localisation de l'utilisateur		Affichage de la carte et de marqueurs	Déploiement sur plusieurs appareils (smartphones)	
Outils et packages utilisables	react-native-community/geolocation	expo-location	react-native-maps (+ clé api Google Maps)	app Expo Go (via un QR code)	page web (via un lien url)
Technologies utilisées					
Managed workflow (expo cli) 	NON	OUI	OUI	OUI	OUI
	Fonctionnel		Fonctionnel	Fonctionnel	
Bare workflow (react-native cli) 	OUI	NON	OUI	NON	NON
	Fonctionnel		Fonctionnel	Non fonctionnel	
App web (react-native web) 	NON*	NON	NON**	NON	OUI
	Non fonctionnel		Non fonctionnel	Fonctionnel	

* l'adresse ip semble poser problème et se localise aux Etats Unis, même en réglant manuellement la localisation de l'android device généré par Android Studio

** l'affichage de la carte pose problème sur le web (non résolu)

Ce comparatif permet de visualiser mes choix techniques et de comprendre pourquoi j'ai finalement créé et déployé mon application avec Expo cli (après avoir finalement découvert le package expo-location), puisque c'est effectivement la seule technologie sur les 5 tests réalisés qui m'a permis de développer mes 3 fonctionnalités principales.

En effet, cette solution me permet bien :

1. d'accéder à la géolocalisation de l'utilisateur
2. d'afficher la carte et des marqueurs (notamment celui de la position du membre)
3. de déployer sur plusieurs appareils pour que tous les membres de l'équipe puissent se localiser et visualiser les autres membres sur la carte (en scannant simplement le QR code grâce à l'application Expo Go)

Travail réalisé

Partie 1 : Modèle de données et api web

Pour ce premier projet (back-end constitué de mon modèle et de mes api web), j'ai réalisé :

- le code du modèle (classes métiers)
- le mapping de la BDD (automatique grâce à Entity Framework Core)
- initialisation et remplissage de la BDD avec des données entrées en dur avec un SeedData.cs
- les api principales :
 - User : get all, get(id) qui est également utilisé pour l'authentification, post(id), delete(id), put(id, longitude, latitude)
 - Mission : get all, get(id)
- une partie des api secondaires :
 - User : post(id), delete(id)
 - Mission : post(id)

Structure de la BDD et données de la table Enrollment :

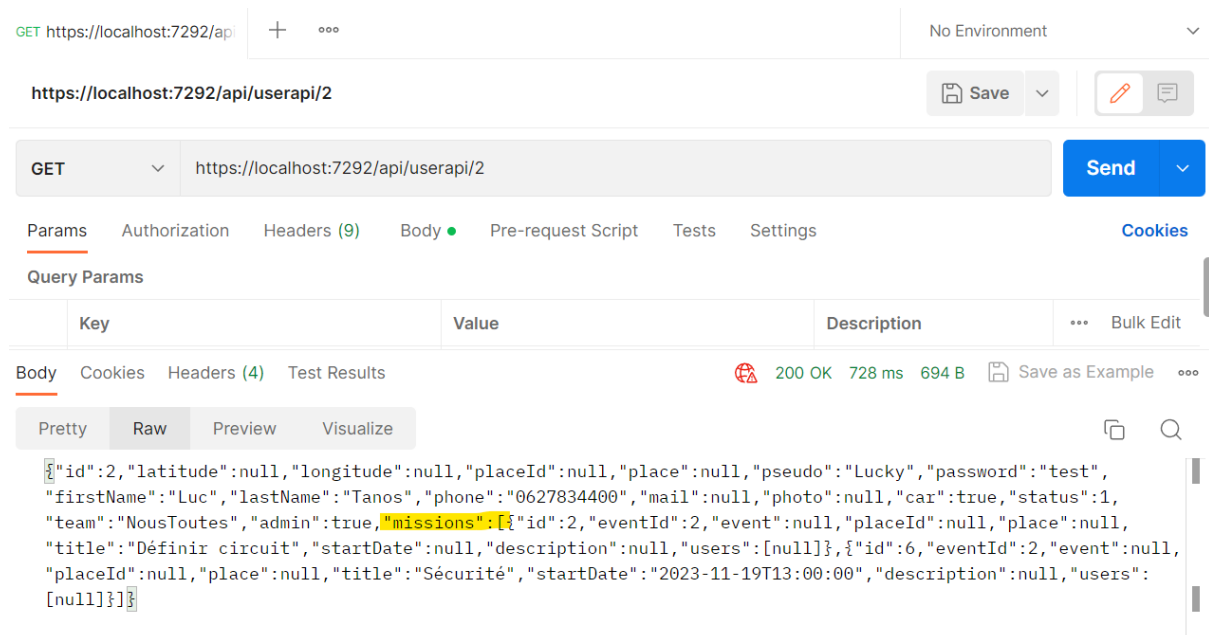
Nom	Type	Schéma
Tables (8)		
Enrollments		CREATE TABLE "Enrollments" ("Id" INTEGER NOT NULL CONSTRAINT "PK_Enrollments" PRIMARY KEY AU
Events		CREATE TABLE "Events" ("Id" INTEGER NOT NULL CONSTRAINT "PK_Events" PRIMARY KEY AU
MissionUser		CREATE TABLE "MissionUser" ("MissionsId" INTEGER NOT NULL, "UsersId" INTEGER NOT NULL
Missions		CREATE TABLE "Missions" ("Id" INTEGER NOT NULL CONSTRAINT "PK_Missions" PRIMARY KEY AU
Places		CREATE TABLE "Places" ("Id" INTEGER NOT NULL CONSTRAINT "PK_Places" PRIMARY KEY AU
Users		CREATE TABLE "Users" ("Id" INTEGER NOT NULL CONSTRAINT "PK_Users" PRIMARY KEY AU
EFMigrationsHistory		CREATE TABLE "EFMigrationsHistory" ("MigrationId" TEXT NOT NULL CONSTRAINT "FK_EF
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Index (6)		
IX_Enrollments_MissionId		CREATE INDEX "IX_Enrollments_MissionId" ON "Enrollments" ("MissionId")
IX_Enrollments_UserId		CREATE INDEX "IX_Enrollments_UserId" ON "Enrollments" ("UserId")
IX_MissionUser_UsersId		CREATE INDEX "IX_MissionUser_UsersId" ON "MissionUser" ("UsersId")
IX_Missions_EventId		CREATE INDEX "IX_Missions_EventId" ON "Missions" ("EventId")
IX_Missions_PlaceId		CREATE INDEX "IX_Missions_PlaceId" ON "Missions" ("PlaceId")
IX_Users_PlaceId		CREATE INDEX "IX_Users_PlaceId" ON "Users" ("PlaceId")
Vues (0)		
Déclencheurs (0)		

Table : Enrollments							
	Id	UserId	MissionId	Accepted	Started	Ended	NeedHelp
1	3	6	18	1	0	0	0
2	4	9	18	0	0	0	0
3	5	8	18	0	0	0	0
4	6	10	17	1	1	0	1
5	7	7	14	0	0	0	0
6	8	2	2	1	1	1	0
7	9	2	6	0	0	0	0
8	10	4	4	0	0	0	0
9	11	4	1	1	0	0	0
10	14	3	1	1	1	0	1
11	15	3	6	1	0	0	0
12	16	1	5	0	0	0	0

Ces figures sont produites à partir du gestionnaire de base de données DB Browser pour SQLite.

On voit bien sur la figure de droite l'intérêt de la table *Enrollment*. Bien qu'elle fasse doublon avec la table *MissionUser*, elle possède en effets les champs supplémentaires "accepted", "started", "ended" et "needHelp" précédemment cités et qui permettent à l'utilisateur de signifier s'il a accepté la mission qui lui est assignée, s'il l'a commencé etc. C'est au travers de l'api PUT décrite plus haut que l'utilisateur va pouvoir modifier la valeur de ces booléens (1 pour "oui" et 0 pour "non").

Exemple d'utilisation de l'api get(id) de user avec l'outil PostMan :



On voit bien (surligné en jaune), que l'api qui renvoie un utilisateur en particulier (donné par son identifiant, ici =3) renvoie également la liste des missions qui lui sont assignées.

Partie 2 : Application mobile

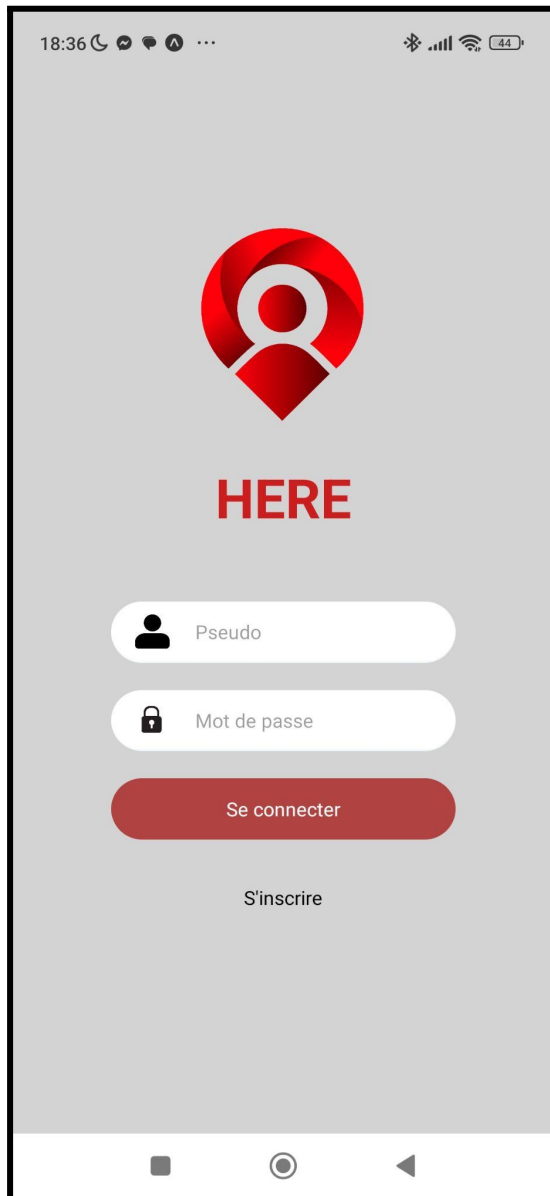
Pour la seconde partie, j'ai pu réalisé :

- l'écran de connexion et reconnaissance de l'utilisateur (*ConnexionScreen*)
- l'écran d'affichage de la carte de la position de l'utilisateur (*UsersMapScreen*)
- l'écran d'affichage de la liste des utilisateurs (*UsersListScreen*)

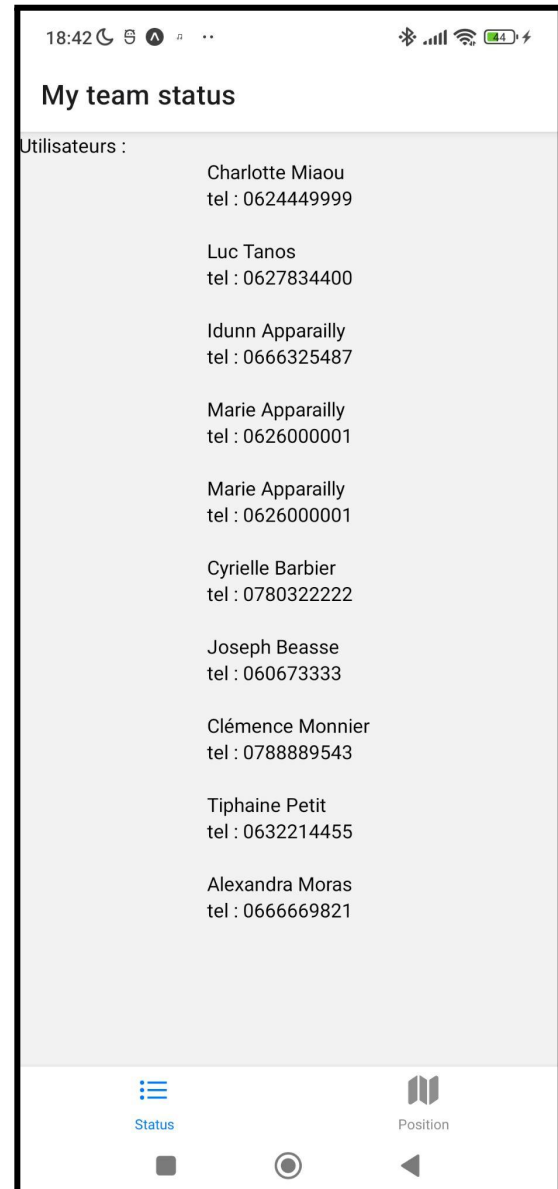
On peut naviguer entre ces différents écrans grâce aux fichiers de navigation (*RootTabNavigator*, *UserStackNavigator* ...), et ces écrans font appel à de nombreux autres

fichiers : api de la première partie du projet (*userService*, *authentication*), api Google map, composants (*userItem*, *userList*), assets etc.

Visuels des différents écrans de l'application (capturés sur mon smartphone) :



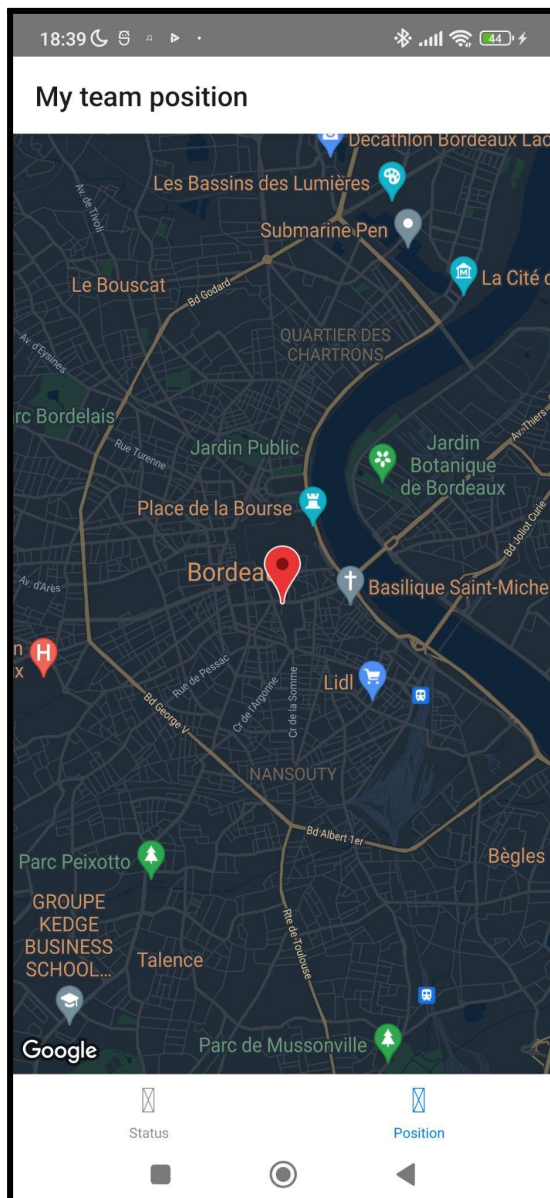
ConnexionScreen.js



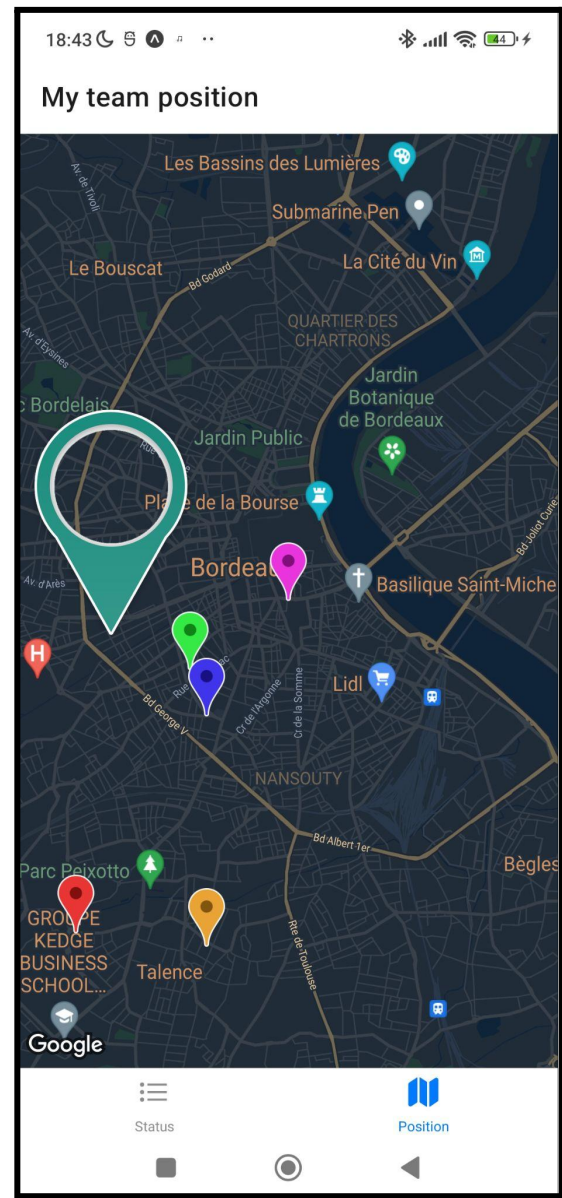
UsersListScreen.js

Une fois que l'utilisateur s'est authentifié, il arrive directement sur l'écran *UsersListScreen* qui liste les membres de l'équipe.

Grâce aux fichiers de navigation (*RootTabNavigator* et *UsersStackNavigator*), il peut également naviguer vers l'écran *UsersMapScreen*.



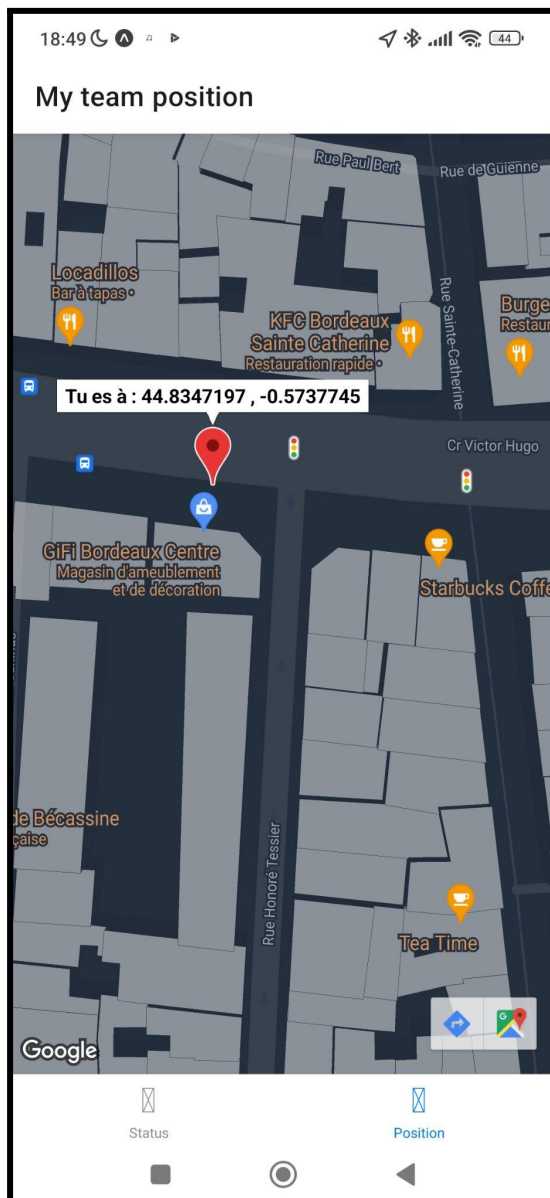
UsersMapScreen.js



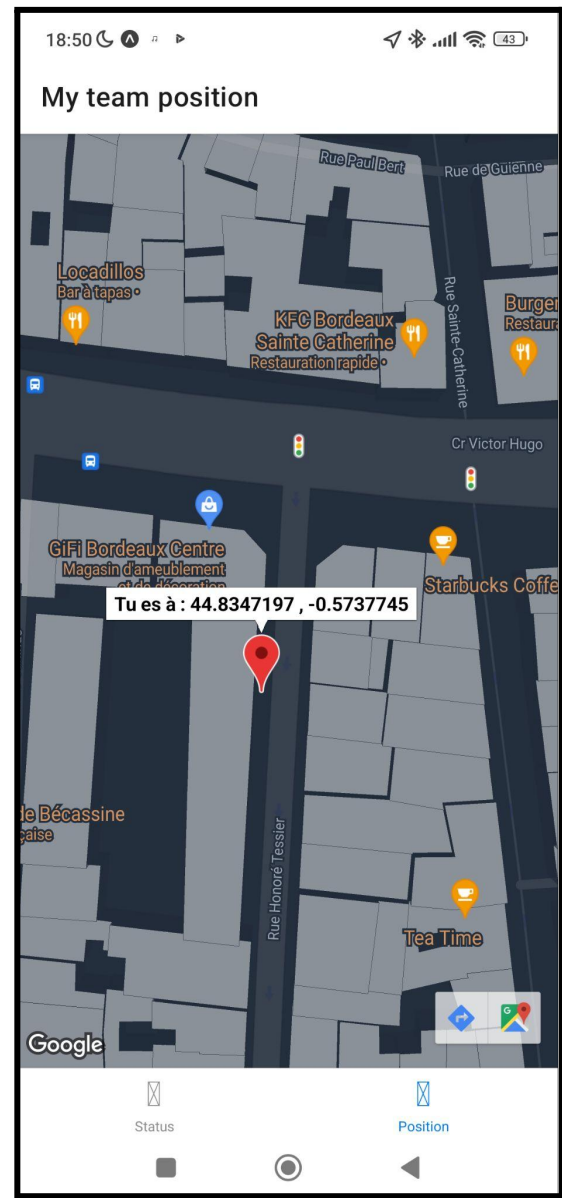
UsersMapScreen.js

L'écran de gauche représente l'écran tel qu'il apparaît lorsque l'utilisateur navigue vers cet écran (et après avoir accepté un message d'alerte de demande de permission d'accès à sa localisation). Initialement, la carte est centrée sur la géolocalisation de l'utilisateur (représentée par le marqueur rouge).

L'écran de droite a été réalisé en créant artificiellement des marqueurs (avec des données entrées en dur dans le code) censés représenter les différents membres de l'équipe.



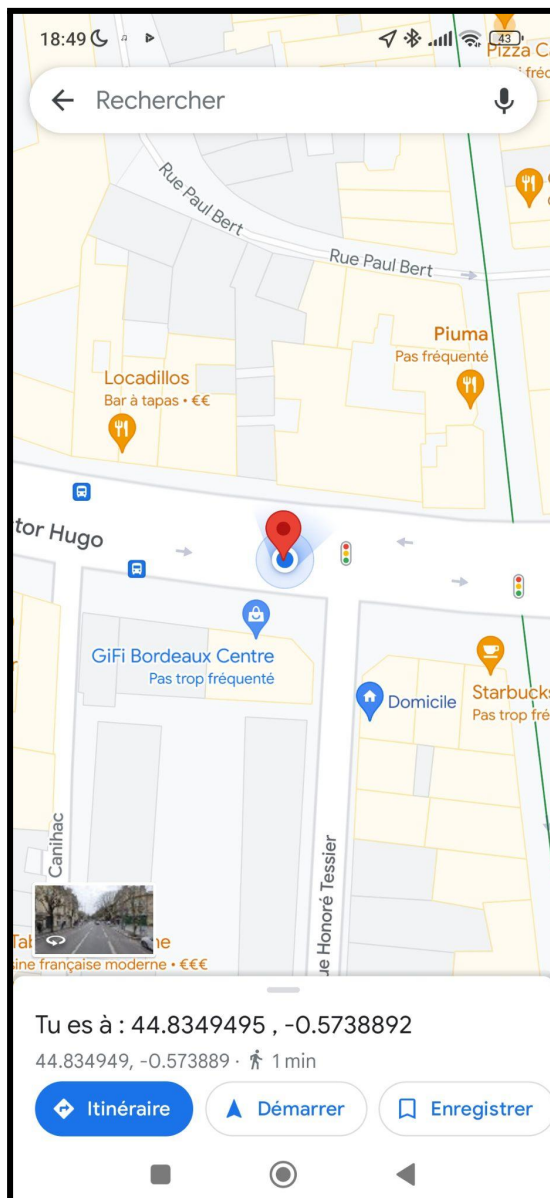
UsersMapScreen.js



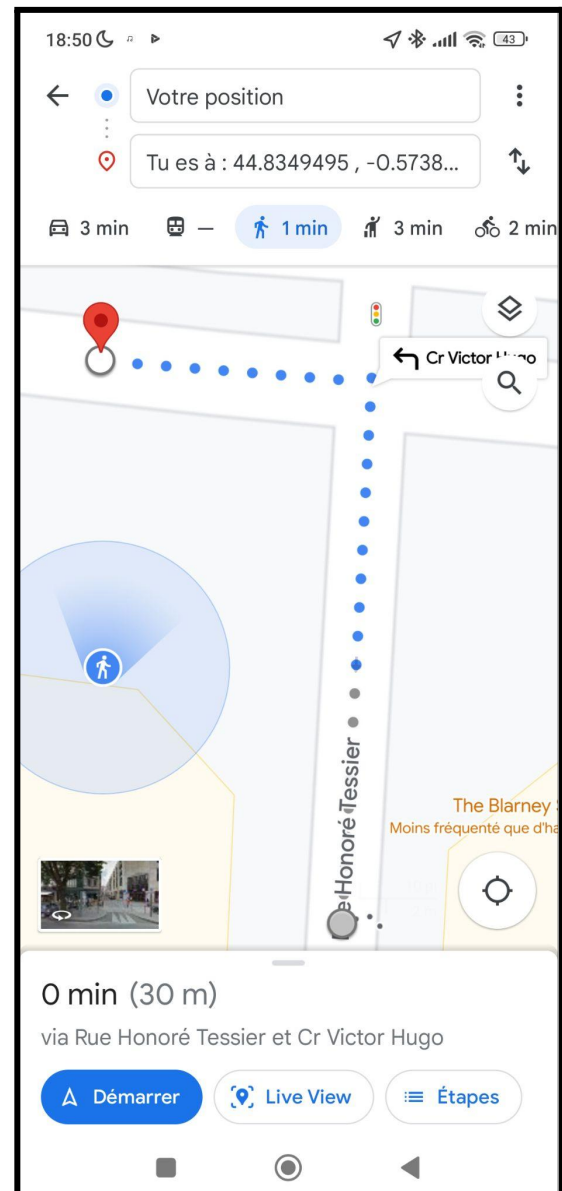
UsersMapScreen.js

L'utilisateur peut ensuite se déplacer et zoomer sur la carte. Ici j'ai zoomé jusqu'à voir mon emplacement réel. On voit bien entre l'écran de gauche et de droite que la localisation se met à jour lorsque l'utilisateur se déplace. La précision en termes d'emplacement est très bonne (on voit que je change de pièce quand je suis dans mon appartement par exemple) et la mise à jour en temps réel également (seulement 2 ou 3 secondes de décalage pour une précision au mètre près).

Lorsque l'on clique sur le marqueur, j'ai créé un message d'alerte qui indique les coordonnées exactes (latitude et longitude) et on voit bien qu'effectivement : elles se modifient et se mettent bien à jour entre les deux captures d'écran.



Ouverture de l'app Google Maps



Itinéraire sur Google Maps

Enfin en cliquant sur l'icône de Google Maps présent en bas à droite de l'écran précédent, l'application nous redirige automatiquement vers l'application Maps. La localisation s'affiche toujours sur Maps et le message d'alerte codé en JavaScript s'affiche et fait office de titre de lieu : "Tu es à : 44.8349495, -0.5738892".

On peut utiliser toutes les fonctionnalités de Google Maps (grâce à la clé d'api que j'ai généré), notamment pointer un lieu et demander un itinéraire depuis sa position actuelle comme on le voit sur l'écran de droite.

Conclusion

Bilan fonctionnel et pistes d'évolution

Au cours de ce projet, j'ai pu réalisé les fonctionnalités principales de mon application mobile, à savoir :

- les api web de user et missions : afin de récupérer les utilisateurs et leurs missions et d'authentifier un utilisateur en particulier
- l'authentification de l'utilisateur
- l'affichage de la liste des membres de l'équipe de l'utilisateur
- la permission d'accès à la localisation de l'utilisateur et sa récupération
- l'affichage de la carte et d'un marqueur montrant la position de l'utilisateur
- la redirection automatique vers l'application Google Maps
- le déploiement via Expo Go afin de pouvoir tester l'application sur plusieurs smartphones et simuler une équipe sur le terrain

Il serait néanmoins pertinent de continuer le projet afin d'améliorer l'application et d'ajouter ces fonctionnalités importantes :

- terminer l'api de modification de l'utilisateur afin de mettre à jour ses coordonnées (latitude et longitude) et ainsi de les stocker dans la base de données afin d'avoir accès à la position des autres utilisateurs connectés
- affichage des marqueurs de position des autres membres de l'équipe
- affichage de la liste des missions
- affichage de la liste des missions assignées à l'utilisateur connecté
- possibilité pour l'utilisateur de modifier le statut de la mission en codant l'api agissant sur les tables *Enrollment*, *User*, *Mission* et *UserMission* : l'accepter, la commencer, la terminer ou demander de l'aide

Pour conclure sur l'aspect informatique du projet, de nombreuses fonctionnalités restent à implémenter avant d'obtenir une application viable et réellement utile, et le front-end mériterait d'être beaucoup plus enrichi.

Bilan personnel

J'ai toujours su que les développeurs passaient beaucoup de temps à lire de la documentation sur internet. Mais ce projet m'a permis de m'en rendre compte de manière très frappante ! J'avais en effet escompté développer de nombreuses fonctionnalités et coder un joli front-end mais j'ai finalement passé 95% de mon temps à installer et désinstaller des packages et mon environnement de manière globale, à gérer des erreurs et incompatibilités et, bien sûr, à lire de la documentation.

Ce projet m'a néanmoins permis d'explorer de nouvelles technologies non abordées en cours, en plus de consolider mes compétences techniques sur celles préalablement étudiées. Surtout, au-delà des compétences de programmation, j'ai acquis des connaissances beaucoup plus solides en termes d'informatique au sens large concernant notamment : le fonctionnement du hardware de ma machine, l'architecture et design des applications, les fonctionnalités de différentes librairies et packages et le versionnage de projet. J'ai en effet beaucoup évolué sur mon utilisation de git et GitHub et ai appris de mes (nombreuses) erreurs.

Bien que frustrée de ne pas avoir pu développer toutes les fonctionnalités que je souhaitais, je suis donc satisfaite de me sentir monter en compétences informatiques grâce à cette expérience très réaliste et enrichissante.

Enfin, j'ai beaucoup appris sur la gestion de projet.

Réaliser un projet individuel ne m'était pas arrivé depuis de nombreuses années (à l'école d'Architecture, presque l'intégralité des projets étaient aussi réalisés en équipe) et j'ai appris à m'organiser de manière beaucoup plus efficace pour répartir les tâches sur le long terme sans compter sur des collègues pour m'imposer des deadlines par exemple.

En outre, le fonctionnement mis en place avec mon tuteur m'a permis de me familiariser avec GanttProject qui, bien que peu esthétique, est très pratique pour le suivi de l'avancement des tâches et surtout la mise à jour du planning. Les points d'avancement hebdomadaires m'ont également permis de mieux m'organiser et de gérer mon temps afin d'éviter de passer de trop longues semaines sur des problèmes techniques par exemple en étant persuadée de pouvoir rattraper mon retard par la suite (ce qui malheureusement n'est jamais réaliste).

En ce qui concerne l'avenir de ce projet, je souhaite poursuivre la création de cette application mobile qui me tient à cœur et me semble très pertinente et utile pour les équipes de terrain.

En plus de poursuivre le développement afin d'implémenter les fonctionnalités et d'améliorer le front-end, j'aimerais également utiliser mes compétences en CCU (Conception Centrée Utilisateurs) afin d'enquêter auprès des utilisateurs ciblés et ainsi créer une application proposant la meilleure expérience utilisateur possible.

C'est dans cette optique que j'ai décidé de contacter l'incubateur de projet *Sit'Innov* à destination des étudiants auto-entrepreneurs, afin de me renseigner et potentiellement me faire accompagner sur les possibilités de poursuite et de développement de ce projet.

Mon projet ne s'arrête donc pas là, et j'espère bien mener **Here** le plus loin possible !