

École polytechnique de Louvain

HaïtiWater 2.0

Evolution de l'application HaïtiWater vers une application entière fonctionnelle hors-ligne

Auteur: **Vincent GRADZIELEWSKI**
Promoteurs: **Kim MENS, Sandra SOARES-FRAZÃO**
Lecteurs: **Celine Deknop, Sebastien Strebelle**
Année académique 2020-2021
Master [120] en sciences informatiques

Table des matières

Résumé	2	5.3 Client	36
Remerciements	3	5.3.1 Stratégie de syn-	
Glossaire	4	chronisation des	
1 Introduction	6	pages	37
2 Contexte	9	5.3.2 Stratégie de syn-	
2.1 Gestion de l'eau en Haïti .	9	chronisation de la	
2.2 Introduction à l'application	10	DB	39
2.2.1 Eléments de l'ap-		5.3.3 Gestion des chan-	
plication	10	gements d'utilisa-	
2.2.2 Utilisateurs de		teurs	42
l'application	11	5.3.4 Sytème de notifi-	
2.2.3 Module "Accueil" .	12	cations	42
2.2.4 Module "Réseau" .	12	5.4 Interface utilisateur	42
2.2.5 Module "Carte" . .	13	5.4.1 Changement de	
2.2.6 Module "Gestion		mode	43
de zone"	14	5.4.2 Module à synchro-	
2.2.7 Module "Historique"	15	niser	44
2.2.8 Module "Rapports"	16	5.5 Serveur	45
2.2.9 Module "Consom-		6 Validation	48
mateurs"	17	6.1 Vérifications automatiques	48
2.2.10 Module "Finances"	18	6.2 Vérifications utilisateurs	
2.3 Problèmes réseaux	19	réels	49
3 Organisation	21	7 Améliorations futures	52
3.1 Approche de travail	21	7.1 Suite du projet	52
3.2 Méthodologie	22	7.2 Défis rencontrés	53
4 Analyse des besoins	24	7.3 Propositions	54
4.1 Besoins fonctionnels	25	8 Conclusion	55
4.2 Besoins non-fonctionnels .	29	A Diagramme de Gantt	57
4.3 Structure des données		B Documents de validation	59
hors-ligne	30	B.1 Questionnaire	59
5 Implémentation	32	B.2 Scenario 1 - Gestion hors-	
5.1 Application de base	32	ligne	61
5.2 Choix technologiques	34	B.3 Scenario 2	62
5.2.1 Choix définitif	36	B.4 Scenario 3	63
		B.5 Scenario 4	63
		B.6 Scenario 5	64

Résumé

Ce travail de fin d'études a été réalisé dans le cadre de mon Master en Sciences Informatiques à l'École Polytechnique de Louvain durant l'année académique 2020-2021 en partenariat avec l'ONG Protos.

Dans ce mémoire, je vais vous présenter mon travail qui consistait à reprendre l'application HaïtiWater développée précédemment par Adrien Hallet, Céline Deknop et Sébastien Strebelle; "La gestion du réseau de distribution d'eau potable en Haïti" [3]. Le but de ce travail est de transformer cette web-application pour qu'elle soit entièrement utilisable **lorsque la connexion au serveur n'est pas disponible**.

Cette application a pour but la gestion du réseau de distribution d'eau potable en Haïti. Je commencerai par présenter le contexte haïtien et les raisons pour lesquelles l'évolution de cette application était nécessaire. Ensuite je présenterai le principe des Progressive Web App (PWA) et pourquoi j'ai choisi d'utiliser cette technologie plutôt qu'une autre. J'aborderai par après la validation de l'application et les feedback que j'ai reçus des utilisateurs. En guise de conclusion, je dresserai une liste des améliorations possibles pour le futur développement de l'application.

Tout le code source de l'application est disponible à cette adresse : <https://github.com/exavince/HaitiWater>

Si vous désirez tester l'application, il suffit de demander un login à un collaborateur et de vous rendre sur le lien suivant <https://haitiwater.sipr.ucl.ac.be> afin de vous connecter à l'application.

Remerciements

Dans le cadre de mon mémoire, j'ai été aidé par différentes personnes sans lesquelles la réalisation de celui-ci n'aurait pas été possible. Je souhaite donc consacrer ces quelques lignes pour les remercier.

Merci aux professeurs Kim Mens et Sandra Soares-Frazão pour leur aide, leur soutien et les conseils indispensables qu'ils m'ont apportés tout au long de ce travail.

Merci à l'ONG Protos qui a amené ce projet.

Merci à Celine Deknop et Sebastien Sterbelle pour l'aide et les conseils qu'ils m'ont apportés à différents moments de ce projet.

Merci à Nahomie Labonte pour l'aide apportée au début du projet afin de bien mettre en évidence les besoins de l'application.

Merci à tous les utilisateurs qui ont pris du temps pour tester l'application et me faire part de leur feedback.

Merci à Thomas Van der Sypt pour ces éclaircissements d'un point de vue rédactionnel.

Merci à mes parents et à mes amis qui m'ont soutenu et m'ont motivé tout au long de mon mémoire.

Glossaire

API	Une API est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications..
ARES-CDD	L'ARES est la fédération des établissements d'enseignement supérieur de la Fédération Wallonie-Bruxelles..
Bootstrap	Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web..
Citizen Science	Participation active des citoyens dans la récupération des informations utilisées dans l'application..
Django	Django est un framework Python de haut niveau, permettant un développement rapide de sites internet, sécurisés, et maintenables..
Framework	Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.
full-stack	Développement à la fois sur le frontend et le backend.
JSON	Format de données textuelles qui permet de représenter l'information sous forme de texte.

Middleware	Logiciel tiers qui crée un réseau d'échange d'informations entre différentes applications informatiques.
Module	Une partie de l'application reprennant.
Moscow	La méthode MoSCoW est une technique visant à prioriser des besoins ou des exigences en matière d'assistance à maîtrise d'ouvrage et de développement logiciel..
Object relational mapping	Les ORM sont des frameworks qui, comme l'indique leur nom, permettent de créer une correspondance entre un modèle objet et un modèle relationnel de base de données..
ONG	Organisme financé essentiellement par des dons privés, qui se consacre à l'action humanitaire..
ORM	Object relational mapping.
PostGis	Extension de base de données spatiale pour postgreSQL.
PostgreSQL	PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant et open source.
PWA	Progressive Web App.
SGBD	Système de Gestion de Base de Données.
UEH	Université d'Etat d'Haïti.

Chapitre 1

Introduction

Contexte

Ce mémoire appartient à un projet de développement financé par ARES-CDD avec quelques partenaires tels que Protos, l’UCL et l’Université d’Etat d’Haïti (UEH). L’objectif de ce projet est de réaliser un système logiciel pilote pour la gestion de la distribution d’eau potable en zone rurale. C’est pour cette raison que l’ONG Protos est active en Haïti depuis plusieurs années.

Protos est une ONG qui a pour but d’améliorer l’accès à l’eau potable dans plusieurs pays du monde afin de les aider à se développer. A la suite de nombreuses crises politiques et catastrophes naturelles qui ont détruit beaucoup d’infrastructures locales, l’accès à l’eau potable est devenu difficile en Haïti. De plus, des incertitudes politiques entravent la reconstruction de ces installations et les populations ne sont pas toujours aidées par les services publiques pour assurer la distribution de l’eau.

En effet, aucune gestion centralisée et organisée par l’Etat n’existe pour les zones rurales éloignées des grandes agglomérations. Des réseaux existent, constitués de points de prélèvement d’eau, de conduites de distribution d’eau et de fontaines situées dans les villages, mais la gestion publique de ceux-ci n’est pas opérationnelle.

L’application créée précédemment [3] propose un appui à ces organismes locaux afin de mieux organiser cette distribution.

Problématique

Actuellement, HaïtiWater est une web-application prévue pour fonctionner uniquement lorsque le réseau est stable et fonctionnel. Malheureusement, en Haïti, la connexion au réseau internet est assez instable, voire indisponible. Il faut donc trouver une solution pour que cette application puisse être utilisable et fiable dans les conditions du contexte haïtien.

Motivations

J’ai réalisé ce travail dans le cadre de mon Master en Sciences Informatiques. L’objectif est de mettre en application les différentes compétences acquises durant mes années d’étude sur un projet de grande envergure.

Au-delà de mettre mes compétences en pratique, ce travail va me permettre d’acquérir de l’expérience dans le développement d’applications et surtout dans l’utilisation des nouvelles technologies du web. Il va me permettre d’apprendre à m’intégrer dans un projet de grande ampleur, de me familiariser avec un environnement logiciel déjà existant et d’y apporter ma touche personnelle.

Le plus motivant dans l’évolution de cette application était de savoir qu’il s’agit d’un projet réel avec de vrais acteurs qui vont utiliser cette application sur le terrain une fois que celle-ci sera terminée.

Objectifs

L’objectif de mon mémoire est de proposer une nouvelle version de l’application HaïtiWater afin que celle-ci soit plus adaptée à un usage sur le terrain. Normalement, cette nouvelle version devrait être déployée en Haïti cette année et ce sont les acteurs locaux qui devraient reprendre sa maintenance et son évolution.

Le second objectif est de délivrer une application propre et stable qui donnera envie aux utilisateurs travaillant sur le terrain de remplacer leurs outils actuels par ce nouvel outil.

Approche

Avant de lancer le développement de cette deuxième version de l'application, un travail d'analyse conséquent a été réalisé afin de déterminer les différentes fonctionnalités à implémenter et la technologie à utiliser. Pour ce faire, j'ai eu recours à l'analyse de Moscow [4].

Après cette phase d'analyse, je me suis lancé dans la phase de développement durant laquelle j'ai implémenté toutes les fonctions décrites lors de la phase d'analyse. Etant donné le nombre de fonctions à implémenter et la nécessité de faire de la documentation pour les futurs développeurs, cette phase a été la plus longue à réaliser.

Pour finir vient la phase de validation. L'application a été présentée à des utilisateurs (certains ayant utilisés la première version, d'autres non) afin qu'ils puissent fournir le feedback nécessaire à son évolution.

Contribution

Grâce au travail réalisé, Protos et les acteurs locaux haïtiens vont pouvoir déployer l'application et l'utiliser sur le terrain, même dans les zones les plus rurales. En effet, celle-ci peut maintenant être utilisée lorsque le réseau n'est plus disponible. Pour l'instant, ces acteurs utilisent des outils basiques tels que le crayon et le papier pour dresser leurs rapports sur le terrain. Cette outil ne permet pas un accès facile aux informations passées et complique la transmission de l'information à la hiérarchie. L'utilisation de l'application permettra de résoudre ces deux problèmes.

J'espère avoir apporté ma contribution à une meilleure gestion de l'eau en Haïti en aidant à l'évolution de cette application. Pour ma part, ce travail m'aura permis d'acquérir de nouvelles compétences dans le développement full-stack d'applications web.

Chapitre 2

Contexte

2.1 Gestion de l'eau en Haïti

Haïti est l'un des pays les plus pauvres au monde. Il est situé dans une zone géographique où les risques de catastrophes naturelles sont très élevés. Ces intempéries détruisent les infrastructures et empêchent le bon développement du réseau de distribution d'eau potable. Il y a quelques années, en 2010, le pays a subi un énorme séisme qui a ravagé une bonne partie du territoire. Parmi tous les défis à relever vient celui de la gestion et de la distribution de l'eau potable, surtout dans les zones les plus rurales. Le climat de la région rendant excessivement difficile l'exploitation directe des cours d'eau, il faut beaucoup d'infrastructures afin de pouvoir assumer la distribution de l'eau potable à tous.

En raison de la grande pauvreté qui règne sur la plupart de l'île et de gros problèmes organisationnels, il est très difficile de maintenir et de développer le réseau de distribution d'eau potable. Il y a un manque significatif de collaboration entre les entités haïtiennes ou entre les villages. Ce manque est dû, en partie, à la communication non optimisée, quasi inexistante dans certains cas. Dans les zones les plus rurales de l'île, le taux de recouvrement des factures est excessivement faible, jusqu'à 11%.

Pour toutes ces raisons, l'ONG Protos vient donc en aide à Haïti afin d'aider le service national des eaux dans la gestion des infrastructures et de la facturation des clients, surtout dans les zones les plus rurales.

Pour de plus amples informations sur les problèmes environnementaux, politiques, sociaux ou organisationnels du contexte haïtiens, je vous invite à consulter le mémoire Céline Deknop, Adrien Hallet, Sébastien Strebelle [3].

2.2 Introduction à l'application

Dans le cadre de la situation décrite dans la section 2.1, Protos a fait appel à l'UCL afin de créer une application d'aide à la gestion et à la distribution de l'eau potable en Haïti. Une première version de l'application a été développée en 2018-2019 par trois personnes dans le cadre de leur mémoire [3]. Elle comprend différents modules permettant de faciliter la gestion des infrastructures et de gérer les différents consommateurs. L'application est basée sur un principe hiérarchique, la figure ?? démontre ce principe et provient du mémoire de Céline Deknop, Adrien Hallet, Sébastien Strebelle [3].

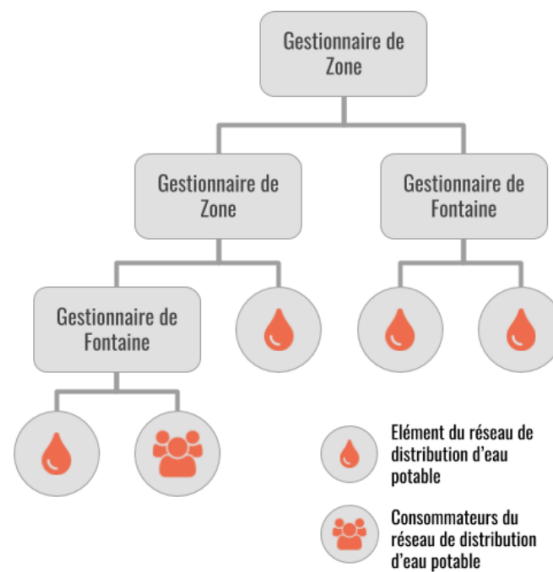


FIGURE 2.1 – Hiérarchie dans l'application [3]

2.2.1 Éléments de l'application

Zone Une zone représente une partie du territoire. Elle peut être subdivisée en plusieurs sous-zones, la zone tutrice reprendra alors toutes les données des zones tutorées. Cela permet de séparer les responsabilités des différents territoires.

Éléments du réseau Ces éléments représentent les points physiques du réseau de distribution d'eau potable. On en dénombre six types : conduites, réservoirs, sources, fontaines, kiosques et prises individuelles. Tous ensemble, ils forment le réseau de distribution au complet.

Consommateurs Le consommateur est une personne qui va utiliser le réseau de distribution d'eau potable. Lors de son enregistrement, chaque consommateur se voit attribué un seul élément de sortie d'eau du réseau : fontaine, kiosque ou prise individuelle. De cette façon, il est plus simple de gérer la facturation des consommateurs.

2.2.2 Utilisateurs de l'application

Il y a deux types d'utilisateurs qui peuvent se connecter à l'application.

Gestionnaire de fontaines Ce gestionnaire va gérer la distribution de l'eau aux différents consommateurs. Il devra gérer tous les éléments du réseau qui lui sont attribués.

Gestionnaire de zones Ce gestionnaire va avoir la responsabilité de gérer d'autres gestionnaires de zones et/ou des gestionnaires de fontaines. Sa tâche principale est d'administrer et de surveiller les personnes qui sont en-dessous de lui.

Ces deux gestionnaires vont avoir des privilèges différents et peuvent donc interagir différemment avec les données. Le tableau 2.1 provenant du mémoire de Céline Deknop, Adrien Hallet et Sébastien Strebelle [3] permet d'illustrer plus simplement cela.

Permission	Gestionnaire	
	fontaine	zone
Ajouter/modifier/supprimer un consommateur	✓	✓
Ajouter/modifier/supprimer un élément du réseau de distribution	✓	✓
Ajouter/modifier/supprimer un rapport mensuel	✓	✓
Ajouter/modifier/supprimer un paiement	✓	✓
Ajouter/modifier/supprimer un ticket de support	✓	✓
Ajouter/modifier/supprimer une zone	✗	✓
Ajouter/modifier/supprimer un gestionnaire	✗	✓*
Accepter/refuser un changement dans l'historique	✗	✓

* Un gestionnaire de zone ne peut pas modifier les informations personnelles (mot de passe, courrier, nom, prénom) d'un gestionnaire existant

TABLE 2.1 – Permissions dans l'application, par type d'utilisateur

2.2.3 Module "Accueil"

Ce module contient les informations condensées de la zone attribuée à l'utilisateur : le nombre de fontaines, de kiosques, de points de prises individuelles, de conduites, de foyers et de consommateurs individuels.

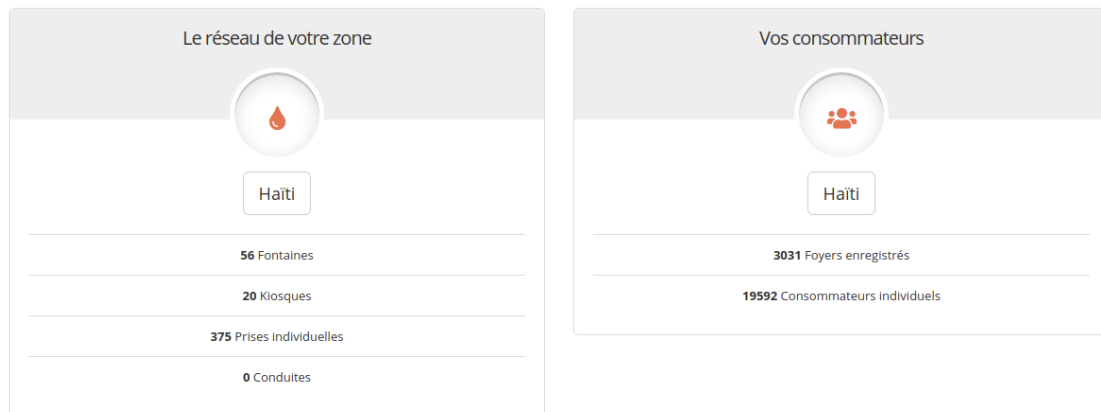


FIGURE 2.2 – Module "Accueil"

2.2.4 Module "Réseau"

Dans ce module, l'utilisateur peut retrouver trois éléments différents.

Schéma Ce schéma représente la répartition des consommateurs selon leur genre ou le volume d'eau mensuel distribué dans chaque zone. L'utilisateur peut sélectionner le schéma à afficher grâce à une liste déroulante.

Résumé de zone Ce résumé présente le nombre de consommateurs et de points d'eau présents ainsi que le volume d'eau distribué dans la zone attribuée à l'utilisateur.

Éléments du réseau Ce tableau reprend tous les différents éléments du réseau. Dans cette partie, en fonction de ses privilèges, l'utilisateur peut ajouter, modifier ou supprimer des éléments du réseau. Il peut également trier ou rechercher les éléments du réseau, si nécessaire.

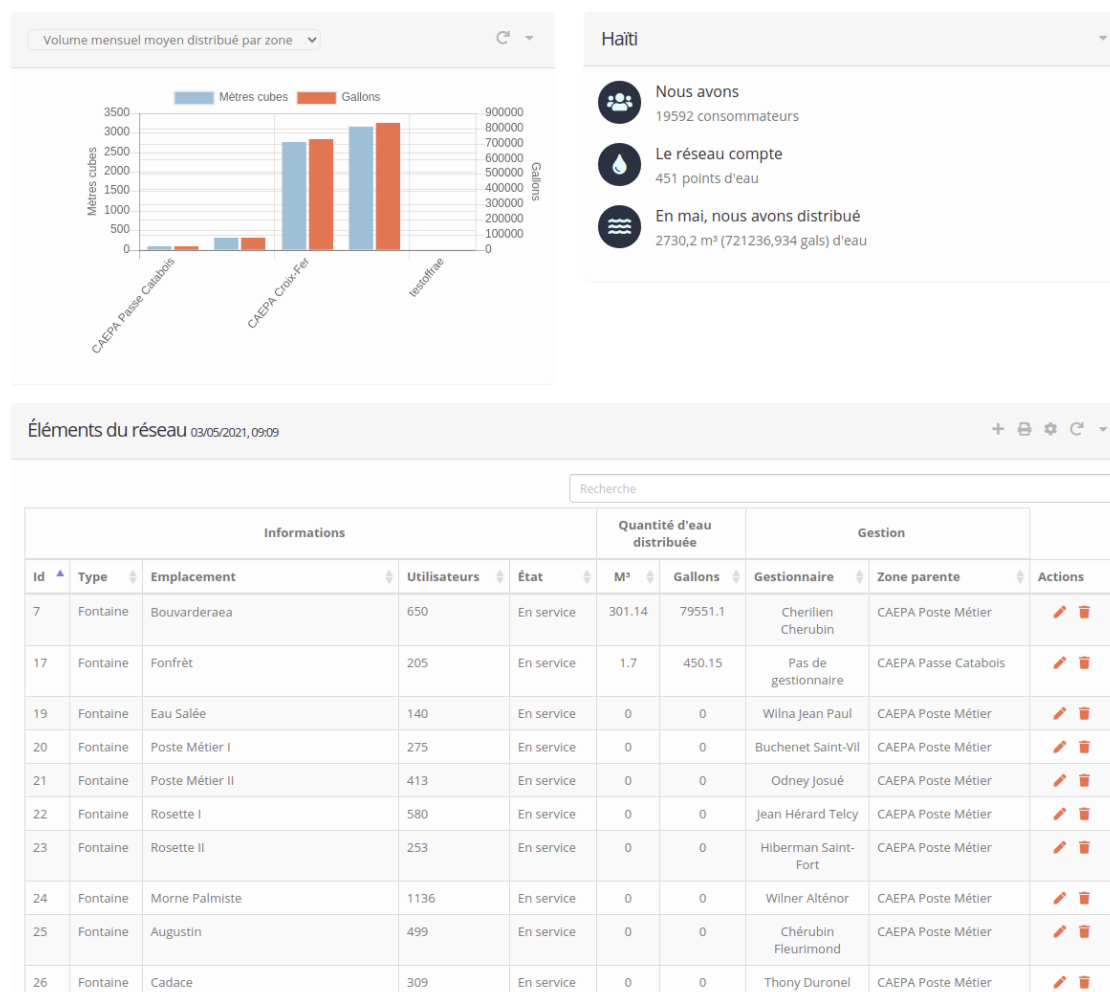


FIGURE 2.3 – Module "Réseau"

2.2.5 Module "Carte"

Ce module comprend un tableau réduit des éléments du réseau ainsi qu'une carte interactive qui permet à l'utilisateur de voir où sont situés les différents éléments du réseau et de connaître ou d'encoder les coordonnées géographiques de ceux-ci. Pour encoder ces coordonnées, l'utilisateur peut soit les entrer manuellement, soit utiliser la carte interactive prévue à cet effet. Comme dans le module décrit au point 2.2.4, l'utilisateur peut ajouter, modifier ou supprimer des éléments du réseau.

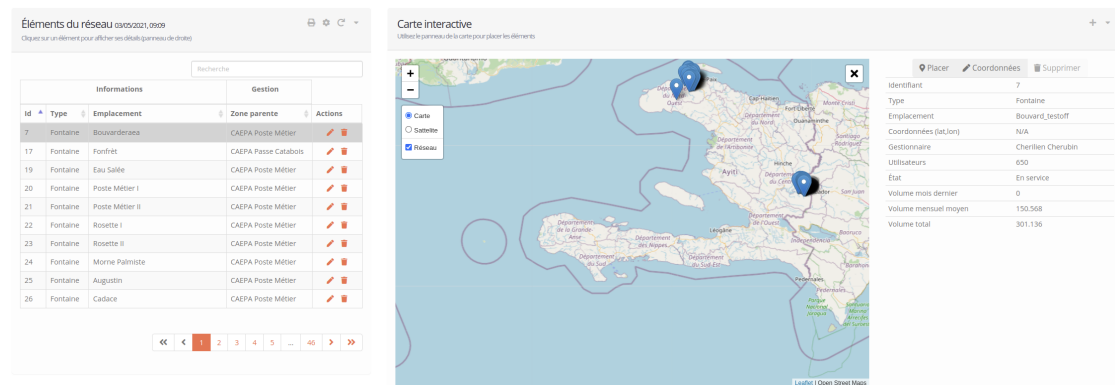


FIGURE 2.4 – Module "Carte"

Éléments du réseau 03/05/2021, 09:09				
Cliquez sur un élément pour afficher ses détails (panneau de droite)				
Recherche				
Informations			Gestion	
ID	Type	Emplacement	Zone parente	Actions
7	Fontaine	Bouvarderaea	CAEPA Poste Métier	[Edit] [Delete]
17	Fontaine	Fonfrêt	CAEPA Poste Métier	[Edit] [Delete]
19	Fontaine	Eau Salée	CAEPA Poste Métier	[Edit] [Delete]
20	Fontaine	Poste Métier I	CAEPA Poste Métier	[Edit] [Delete]
21	Fontaine	Poste Métier II	CAEPA Poste Métier	[Edit] [Delete]
22	Fontaine	Rosette I	CAEPA Poste Métier	[Edit] [Delete]
23	Fontaine	Rosette II	CAEPA Poste Métier	[Edit] [Delete]
24	Fontaine	Morne Palmiste	CAEPA Poste Métier	[Edit] [Delete]
25	Fontaine	Augustin	CAEPA Poste Métier	[Edit] [Delete]
26	Fontaine	Cadace	CAEPA Poste Métier	[Edit] [Delete]

Placer	Coordonnées	Supprimer
Identifiant	7	
Type	Fontaine	
Emplacement	Bouvard_testoff	
Coordonnées (lat,lon)	N/A	
Gestionnaire	Cherilien Cherubin	
Utilisateurs	650	
État	En service	
Volume mois dernier	0	
Volume mensuel moyen	150.568	
Volume total	301.136	

(a) Tableau des éléments réseaux simplifié

(b) Détails de l'élément réseau

2.2.6 Module "Gestion de zone"

Ce module comprend trois tableaux permettant à l'utilisateur de gérer la zone qui lui est attribuée.











Zones Le premier contient les différentes zones géographiques encodées dans le système. Cliquer sur un des éléments du tableau permet de filtrer les éléments des deux autres tableaux décrits plus bas afin de ne garder que les éléments de la zone concernée. Si l'utilisateur a les privilèges requis, ce tableau permet également d'ajouter, modifier ou supprimer une zone.

Gestionnaires Le deuxième contient la liste de tous les gestionnaires. Cliquer sur un gestionnaire permet à l'utilisateur de filtrer les éléments réseaux qui sont gérés par ce gestionnaire. De nouveau, si l'utilisateur possède les privilèges requis, il pourra ajouter, modifier ou supprimer des gestionnaires.

Elements du réseau Le dernier tableau est identique à celui de la section 2.2.4.





















Zones
07/05/2021, 13:00
Cliquez sur une zone pour filtrer les gestionnaires et éléments

Recherche

Id	Nom	Actions
1	Haiti	 
3	CAEPA Poste Métier	 
5	CAEPA Passe Catabois	 
15	CAEPA Croix-Fer	 
36	testoffrae	 

Gestionnaires et Techniciens 03/05/2021, 09:09
Cliquez sur un gestionnaire pour filtrer les éléments

Recherche

Id	Nom	Prénom	Téléphone	Courriel	Rôle	Zone	Actions
Anfas Madam Denaud	Delaud	Bernard	48160978	fag.10.fag@gmail.com	CAEPA Poste Métier		 
Augustin	Fleurimond	Chérubin	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	25	 
BOUVARD	Cherubin	Cherillen	Non spécifié	mwenfri86@gmail.com	CAEPA Poste Métier	7	 
CADACE	Duronel	Thony	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	26	 
CAPORAL	Jean	Hertilus	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	27	 
CROIX-FER	Bocicaut	Romer		louisfritzno@yahoo.fr	CAEPA Croix-Fer		 
Dogad	Jean	Bernado	41577046	fag.10.fag@gmail.com	CAEPA Croix-Fer	192	 
Eau Salée	Jean Paul	Wilna	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	19	 
Ecole Nationale	Belizaire	Marie Itelise	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	45	 
Fond Boulé	Saint-Vil	Michelet	Non spécifié	fritznolouis@gmail.com	CAEPA Poste Métier	28	 

(a) Tableau des zones

(b) Tableau des gestionnaires

2.2.7 Module "Historique"

Ce module contient les actions ayant été effectuées par des gestionnaires de fontaines. Ces actions doivent être traitées (validées ou refusées) par un gestionnaire plus haut placé. On peut y retrouver deux tableaux.

A valider Ce tableau reprend les éléments en attente de traitement : le gestionnaire de zones peut valider ou refuser l'action encodée.

Historique Ce tableau reprend les éléments qui ont été validés ou refusés dans les trois dernières semaines. Il s'agit de l'historique récent des dernières modifications.

Actions effectuées 03/05/2021, 09:09						
Cliquez sur une ligne pour afficher les détails ! Validez ou refusez les changements avec les boutons d'action.						
Id	Horodateur	Type	Utilisateur	Modification	Actions	
3800	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3801	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3802	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3803	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3804	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3805	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3806	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3807	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3808	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3809	2019-05-03	Ajouter	PC	Rapport mensuel	<input checked="" type="checkbox"/>	<input type="checkbox"/>

<< < 1 2 3 4 5 ... 214 > >>

Actions effectuées 03/05/2021, 09:09						
Vous pouvez visualiser les changements acceptés ou validés des trois dernières semaines.						
Id	Horodateur	Type	Utilisateur	Modification	Action choisie	
3271	2019-04-03	Ajouter	PM	Paiement de consommateur	Validé	
3272	2019-04-05	Ajouter	PC	Ticket de problème	Validé	
3762	2019-05-02	Ajouter	PM	Paiement de consommateur	Validé	
3771	2019-05-03	Ajouter	PC	Ticket de problème	Validé	

FIGURE 2.7 – Module "Historique"

2.2.8 Module "Rapports"

Dans ce module, l'utilisateur peut signaler les différents problèmes qu'il rencontre avec les infrastructures de distribution d'eau potable. Le problème signalé, l'utilisateur peut consulter, modifier ou supprimer son signalement. Si ses privilèges sont suffisamment élevés, il pourra également gérer les signalements d'autres utilisateurs.

C'est également ici que l'utilisateur va pouvoir encoder les rapports mensuels. Si jamais la connexion au serveur n'est pas disponible, il peut sauvegarder le formulaire avec les données qu'il a encodées afin de l'envoyer plus tard lorsque la connexion sera rétablie.

Envoyer un rapport

Le rapport sera visible pour tous vos gestionnaires.

Signaler un problème

Rapport mensuel

Tickets de support

03/05/2021, 09:09

⚙️ ↻ ▼

Recherche









Id ▲	Urgence ⚡	Emplacement ⚡	Type ⚡	Commentaire ⚡	Statut ⚡	Actions
26	Moyen	Fontaine Ti Charles	Mécanique	tuyau bobotyrytyrt	Non résolu	 
30	Haut	Fontaine fon Melon	Mécanique	pa gen dlo	Non résolu	 
32	Haut	Kiosque Kote Iegliz katolik	Mécanique	Dlo a pa rive nan fonten yo	Non résolu	 
34	Bas	Prise Individuelle Samuel Etzer	Mécanique	dazf	Non résolu	 

FIGURE 2.8 – Module rapport

2.2.9 Module "Consommateurs"

Dans ce module, l'utilisateur retrouve trois parties différentes.

Schéma Ce schéma est celui de la section 2.2.4.

Résumé de zone Cette partie présente un résumé de la zone attribuée à l'utilisateur (nombre de foyers consommant de l'eau, nombre de consommateurs, nombre de foyers n'ayant pas payé leur facture).

Consommateurs Ce tableau reprend tous les consommateurs appartenant à la zone attribuée à l'utilisateur. Celui-ci peut ajouter, modifier ou supprimer des consommateurs. Il peut également consulter tous leurs les détails.











<div> <div>Consommateurs</div> <div>03/05/2021, 09:10</div> <div> + <div></div> <div></div> <div></div> <div></div> </div> </div>										
<div> <div>Recherche</div> </div>										
Id	Nom	Prénom	Sexe	Adresse	Téléphone	Consommateurs à charge	Adduction Principale	Somme Due	Zone	Actions
12	Abella alcaïl	Chantale	Femme	La Bale	36522698	1	Fontaine Wout Simetyè (La Bale)	-186	CAEPA Poste Métier	 
13	Acceus newone	Assonaise	Femme	Ti Charles		8	Fontaine Ti Charles	-12	CAEPA Poste Métier	 
15	Vilce	Waldeck	Homme	Poste Métier	Non spécifié	3	Prise individuelle Vilce Waldeck	-150	CAEPA Poste Métier	 
17	Alma	Elibert	Homme	Poste Métier	Non spécifié	1	Prise individuelle Alna Elibert	0	CAEPA Poste Métier	 
18	Aplus	Evens	Homme	La Bale	Non spécifié	1	Prise individuelle Aplus Evens	0	CAEPA Poste Métier	 

FIGURE 2.9 – Tableau des consommateurs

2.2.10 Module "Finances"

Dans ce module, l'utilisateur a accès à quatre tableaux différents qui lui permettent de gérer les finances de ses consommateurs. Celui-ci pourra ajouter, modifier ou supprimer des informations dans les différents tableaux en fonction de son niveau de privilèges.

Zones Ce tableau contient les zones attribuées à l'utilisateur. Le fait de sélectionner une zone permet à l'utilisateur de filtrer les consommateurs.

Consommateurs Ce tableau reprend les consommateurs appartenant à la zone attribuée à l'utilisateur. Le fait de sélectionner un consommateur permet de faire apparaître les deux tableaux suivants.

Détails Ce tableau reprend les coordonnées du consommateur sélectionné ainsi que la somme due par celui-ci. Il n'est pas possible de modifier des données dans ce tableau.

Paiements Ce tableau reprend tous les paiements effectués par le consommateur sélectionné. L'utilisateur peut ajouter, modifier ou supprimer des paiements du consommateur sélectionné.

Zones

+

⚙

▼

Recherche

Id	Nom	Actions
1	Haïti	
3	CAEPA Poste Métier	
5	CAEPA Passe Catabols	
15	CAEPA Croix-Fer	
36	testoffrae	

<<

<

1

>

>>

Consommateurs

03/05/2021, 09:10

+

⚙

↺

↻

▼

Recherche

Id	Nom	Prénom	Somme Dûe	Zone	Actions
12	Abella alcaïl	Chantale	-186	CAEPA Poste Métier	
13	Acceus newone	Assonaise	-12	CAEPA Poste Métier	
15	Vilce	Waldeck	-150	CAEPA Poste Métier	
17	Alma	Elibert	0	CAEPA Poste Métier	
18	Aplus	Evens	0	CAEPA Poste Métier	
19	Behomme	Djomil	0	CAEPA Poste Métier	
20	Belhomme	Walner	0	CAEPA Poste Métier	
21	Belizaïre	Richerson	0	CAEPA Poste Métier	
22	Belony	Angelita	0	CAEPA Poste Métier	
23	Belony	Colbert	0	CAEPA Poste Métier	

<<

<

1

2

3

4

5

...

304

>

>>

Détails de Abella alcaïl Chantale

Somme dûe ?

(HTG) -186

Prochaine facturation

Pas de prochaine facturation

Identifiant

12

Nom

Abella alcaïl

Prénom

Chantale

Genre

Femme

Adresse

La Bale

Téléphone

36522698

Consommateurs à charge

1

Adduction principale

Fontaine Wout Simetyè (La Bale)

Païements de Abella alcaïl Chantale

03/05/2021, 09:09

+

⚙

↺

↻

▼

Recherche

Id	Date	Valeur	Actions
9	2021-05-01	74	
11	2019-05-02	100	
72	2021-05-01	12	

<<

<

1

>

>>

FIGURE 2.10 – Module "Finances"

2.3 Problèmes réseaux

L'application HaïtiWater est une application web classique. Cela signifie que pour fonctionner, il faut qu'il y ait nécessairement une connexion au serveur. Or, en Haïti, cette connexion n'est pas très stable en ville et peut aller jusqu'à être inexistante dans les zones les plus rurales.

Ces problèmes de connexion rendent difficile l'utilisation sereine de l'application sur le terrain. Pour pallier ce problème, il a été décidé de faire évoluer l'application web existante de manière à fonctionner lorsque la connexion au serveur est absente.

Au vue de sa nature, une application web nécessite une connexion pour fonctionner. Il a donc fallu étudier les différentes options qui pourraient permettre à l'application de fonctionner hors-ligne. Parmi ces options, les deux retenues étaient la création d'une **application mobile** séparée de l'application web, ou la transformation de l'application web existante en **PWA**.

Chapitre 3

Organisation

Etant le seul étudiant à réaliser ce mémoire, il n'a pas été nécessaire d'utiliser des outils de planification collaboratif. Cette section contiendra surtout la planification des différentes tâches à accomplir permettant l'évolution de l'application ainsi que l'écriture de ce mémoire. Celles-ci ont été mises en place grâce aux discussions avec mes deux promoteurs.

3.1 Approche de travail

La première étape du mémoire a été de mettre en avant les différentes fonctionnalités à implémenter afin d'ajouter de la plus-value au projet existant, l'idée de base étant de rendre l'application fonctionnelle hors-ligne.

Nous avons étudié la question avec mes deux promoteurs et Nahomie, l'étudiante venue de Haïti. Cette dernière nous a apporté son expertise afin de sélectionner les fonctionnalités qui pourraient apporter une réelle valeur au projet et de prioriser l'implémentation de celles-ci.

Planification

Mensuelle D'abord, il a fallu réaliser un plan général de la réalisation du mémoire, mettre en avant les modules sur lesquels il fallait travailler en priorité et les différentes fonctionnalités à implémenter. Une échéance importante était le test de la nouvelle version de l'application par des utilisateurs à partir de février afin d'obtenir un feedback concret et pertinent.

Hebdomadaire Une réunion était prévue toutes les semaines en alternance avec mes deux promoteurs. Ces réunions permettaient de leur montrer l'état d'avancement du développement de l'application et d'avoir un feedback extérieur. Elles ont toutes eu lieu en appel vidéo via l'application Microsoft Teams. En effet, à cause de la conjoncture liée à la crise sanitaire mondiale, il était impossible de tenir nos réunions en présentiel.

Quotidienne Pour gérer l'organisation de mon travail, une liste de tâches était maintenue. Cette liste était modifiée tous les jours en fonction du travail effectué. Les plus importantes étaient maintenues en haut de la liste afin de les prioriser.

Le plan n'a malheureusement pas pu être entièrement respecté pour différentes raisons :

- La technologie employée (voir section 5.2) étant encore assez récente, les sources d'informations et d'aides sur internet sont encore peu fournies. Du retard a donc été pris lors du développement de certaines fonctionnalités complexes,
- L'installation du serveur en Haïti a pris plus de temps que prévu ; il a donc fallu retarder le test de la nouvelle version de l'application par des utilisateurs,
- Le bal des mesures sanitaires a relativement influencé ma motivation et mon implication dans la réalisation de mon travail.

Malgré le retard accumulé, il a tout de même été possible de terminer le projet dans les temps.

3.2 Méthodologie

Même en travaillant seul, une bonne méthodologie de travail permet de garantir l'avancement régulier du projet. Elle permet de transformer efficacement les différents besoins en fonctionnalités implémentées.

Agile

La méthodologie agile permet une réponse au changement plus flexible au changement. Plûtôt que de planifier tout le projet directement, il est conseillé de développer par petit incrément finalisé à chaque itération.

Cette méthode a permis l'évolution du projet en fonction des différents feedback reçus, ce qui n'aurait pas été possible avec une approche waterfall plus séquentielle.

Feature Driven Development Cette méthode s'attarde sur la création la création d'une liste de fonctionnalités et de leur production. Ces différentes fonctionnalités ont été déterminées par mes deux promoteurs, l'étudiante haïtienne.

Itérations Dans le cadre d'une méthodologie agile, les itérations sont de courtes durées. Lors de ce mémoire, les itérations duraient deux semaines. Cela permettait d'avoir un feedback de la part des deux promoteurs que je voyais en alternance une semaine sur deux. Les avantages des itérations courtes sont la détection rapide des défauts présents ainsi que leur correction.

Chapitre 4

Analyse des besoins

Cette phase est très importante. En effet, nous, mes deux promoteurs, l'étudiante de Haïti et moi-même, avons choisi quelle technologie était la plus adaptée afin de faire évoluer l'application existante de la meilleure façon possible. Nous avons également déterminés toutes les fonctionnalités à implémenter et leur ordre de priorité.

Avec l'aide de l'étudiante haïtienne et les bons conseils de mes promoteurs, j'ai réalisé une analyse de Moscow décrite dans la section 4.1. Cette méthode permet de prioriser les tâches d'un projet en fonction de leur criticité, c'est-à-dire du niveau d'impact de la tâche sur la réalisation du projet. Cette analyse de Moscow est assez représentative de la direction que devait prendre le projet et représente les différentes tâches à accomplir de manière fidèle.

4.1 Besoins fonctionnels

Fonctionnalités hors-ligne	M	S	C	W	Notes
Accès à l'application	✓				Indispensable
Afficher les informations de l'utilisateur connecté	✓				Tâche importante mais pas vitale
Modifier son profil				✓	N'impacte pas l'utilisation hors-ligne
Visualiser le volume mensuel distribué par zone		✓			Cela peut être fait sur le long terme
Répartir les consommateurs par genre		✓			Cela peut être fait sur le long terme
Lister <ul style="list-style-type: none"> les éléments du réseau les consommateurs les paiements les tickets les zones les gestionnaires les logs 	✓ ✓ ✓ ✓	 ✓ ✓	 ✓		Indispensable au travail de terrain Facilite le travail de terrain N'est utile qu'aux gestionnaires de zone qui ne sont pas sur le terrain
Afficher les éléments du réseau sur la carte			✓		Cela peut être fait sur le long terme
Ajouter, supprimer ou éditer <ul style="list-style-type: none"> les consommateurs les paiements les tickets les éléments du réseau les zones les gestionnaires les logs 	✓ ✓ ✓	 ✓	 ✓ ✓ ✓		Indispensable au travail de terrain Facilite le travail de terrain N'est utile qu'aux gestionnaires de zone qui ne sont pas sur le terrain
Signaler un problème (soit mécanique, de qualité ou autres) par le gestionnaire de fontaine	✓				Indispensable pour une bonne gestion du service de distribution d'eau
Envoi de notification au gestionnaire de zone une fois un problème signalé				✓	Tâche importante mais inaccessible à cause du coût de développement qui n'est pas compatible avec le temps qui reste.
Faire un rapport mensuel <ul style="list-style-type: none"> Encoder une consommation Mise à jour de l'état de paiement du consommateur 	✓ ✓				Indispensable pour un suivi du service de distribution d'eau Indispensable pour un suivi du service de distribution d'eau

FIGURE 4.1 – Analyse de Moscow des fonctionnalités hors-ligne

Les besoins fonctionnels répondent aux points précis à implémenter et représentent toutes les tâches à accomplir pour que le projet soit réussi. Dans le tableau 4.1, les lettres ont la signification suivante :

- **M** signifie "Must have", le projet serait un échec si l'application ne possédait pas cette fonctionnalité à la fin du développement,
- **S** signifie "Should have", cette fonctionnalité doit être développée dans la mesure du possible. Ces tâches sont importantes mais pas vitales,
- **C** signifie "Could have", cette fonctionnalité peut être développée si cela n'affecte pas les autres tâches plus importantes. Ce sont des tâches de confort qui peuvent être effectuées si le temps le permet et que les tâches des deux catégories précédentes ont été réalisées,
- **W** signifie "Won't have but would like", ce qui ne sera pas développé cette fois, mais plus tard. Réaliser ces tâches est un luxe théoriquement impossible.

Toutes les tâches de type **M**, **S** et **C** ont été réalisées durant ce projet. Seules les tâches **W** n'ont pas été implémentées.

Lors de l'analyse des besoins, nous avons bien entendu repris l'analyse faite par les anciens mémorants [3]. Ils proposaient trois types différents d'accès à l'application et aux données lorsque la connexion au serveur n'est pas disponible. Les trois options proposées sont les suivantes.

Accès aux formulaires Avec cette solution, l'utilisateur aurait un accès permanent aux différents formulaires d'ajout de données. Celui-ci pourrait pré-remplir les formulaires et les sauvegarder dans le cache du navigateur afin de les envoyer plus tard dans le cas où la connexion n'était pas disponible immédiatement. Cette solution est déjà partiellement mise en place dans l'application de base. Il est en effet déjà possible de pré-remplir son rapport mensuel et de le sauvegarder pour plus tard.

Accès statique aux données Cette solution requiert l'accès à une copie simplifiée de la base de données sur l'appareil de l'utilisateur sans possibilité de modification. Cela permettrait aux utilisateurs de consulter les données hors-ligne mais pas d'en entrer de nouvelles. L'avantage de cette solution est que la synchronisation des tables est très simple et qu'il n'y a pas de problèmes d'inconsistances des données qui peuvent arriver lorsque deux utilisateurs modifient la même donnée en étant hors-ligne.

Accès complet aux données Cette solution permettrait aux utilisateurs de consulter, ajouter, modifier et supprimer des données de la base de données. Ainsi, même en cas de perte de connexion au serveur, l'utilisateur pourrait continuer à réaliser ses tâches sans le moindre souci. Cette solution est la plus compliquée à mettre en place à cause de la synchronisation des différentes versions des données. En effet, si deux utilisateurs modifient en même temps les mêmes données en étant hors-ligne, il faudrait pouvoir déterminer quelles modifications conserver sans créer d'inconsistances.

L'option choisie est l'accès complet aux données. C'est la seule solution qui permettrait vraiment à l'application d'être utilisée de manière intensive sur le terrain. C'est également l'option la plus intéressante à mettre en place du point de vue technique.

Affichage des pages

Une partie importante des besoins est bien évidemment l'accès aux différentes pages de l'application lorsque la connexion au serveur n'est plus disponible.

C'était la partie prioritaire du projet ; sans cette fonctionnalité, il aurait été impossible pour l'utilisateur d'accéder aux différentes données. Les premières pages concernées par cette évolution ont été celles utiles aux gestionnaires de fontaines qui sont le plus souvent sur le terrain. Les différents points du tableau 4.1 concernés par cette partie sont :

- Accéder à l'application,
- Afficher les informations de l'utilisateur connecté,
- Visualiser le volume mensuel distribué par zone,
- Répartir les consommateurs par genre.

Nous constatons que les deux premiers points sont bien placés dans la catégorie "Must have". Les deux derniers sont dans le "Should have". En effet, ces fonctionnalités ne sont que des schémas qui s'affichent sur certaines pages et ceux-ci ne sont pas essentiels.

Accès aux données

La deuxième partie importante des besoins fonctionnels est d'avoir accès aux données de la base de données même lorsque la connexion au serveur n'est pas disponible. En effet, le simple accès aux pages hors-ligne sans pouvoir consulter de données n'a que peu d'intérêt. Dans le tableau 4.1, cela est représenté par les points suivants :

- Lister les éléments du réseau,
- Lister les consommateurs,
- Lister les paiements,
- Lister les tickets,
- Lister les zones,
- Lister les gestionnaires,
- Lister les logs.

Les quatre premiers points sont dans la catégorie "Must have". Ceux-ci représentent les données qui seront utilisées en permanence par les utilisateurs sur le terrain, les gestionnaires de fontaines. Ce sont donc les données qu'il faut prioritairement avoir dans l'application hors-ligne.

Les trois derniers sont dans la catégorie "Should have" et "Could have". Ceux-ci sont surtout utiles pour les gestionnaires de zones qui seront assez peu sur le terrain. Malgré tout, il reste intéressant de pouvoir accéder à ces données même lorsque la connexion au serveur n'est pas disponible dans le cas où un gestionnaire devrait se déplacer sur le terrain.

Un dernier point non listé est d'afficher les éléments du réseau de distribution d'eau sur une carte interactive même lorsque l'on a pas de connexion au serveur. Ce point a été mis dans les "Could have" ; l'usage de la carte en mode hors-ligne est une fonction qui prendrait beaucoup d'espace sur un appareil mobile et cette fonctionnalité n'est pas essentielle à un usage sur le terrain.

Modifications des données

La dernière partie mais non la moindre consiste à gérer l'envoi des données lorsque la connexion au serveur n'est pas disponible. Cela permettrait aux utilisateurs de complètement se passer de la solution crayon et papier. Dans le tableau 4.1, cette partie est représentée par les points suivants :

- Ajouter, éditer ou supprimer :
 - Les consommateurs,
 - Les paiements,
 - Les tickets,
 - Les zones,
 - Les gestionnaires,
 - Les logs,
- Signaler un problème par le gestionnaire de fontaines,
- Faire un rapport mensuel.

La plupart des points sont situés dans la partie "Must have". Ces fonctionnalités sont effectivement essentielles au bon fonctionnement de l'application hors-ligne. Par contre, la modification des éléments du réseau, des zones, des gestionnaires et des tickets sont dans les catégories "Should have" and "Could have", ces données étant surtout utilisées par les gestionnaires de zones qui ne vont que peu sur le terrain.

4.2 Besoins non-fonctionnels

Les besoins non-fonctionnels sont soit des besoins optionnels, soit des besoins liés à l'implémentation et à l'interopérabilité générale. C'est en réalisant ces besoins non-fonctionnels qu'est obtenu un système fiable et qualitatif.

Multi-plateforme

C'est dans le cadre de la suite d'un mémoire universitaire que s'inscrit mon projet. A terme, la nouvelle version de l'application sera déployée et maintenue par une équipe de développeurs en Haïti.

Cependant, un problème persiste; le temps et le budget qui seront alloués à la suite de ce projet ne sont pas encore connus. Par contre, il est primordial que l'application puisse tourner sur un maximum d'appareils différents. Les choix technologiques décrits dans la section 5.2 ont donc été pris en tenant compte du fait que l'application doit pouvoir tourner sur tout type de configuration.

Technologies simples et populaires

Comme précisé précédemment, nous ne connaissons pas encore l'équipe de développeurs qui va reprendre le projet. Il a donc fallu continuer à utiliser des technologies qui sont assez rapides à apprendre et accessibles à tous, cela afin de faciliter la maintenance et l'évolutivité de l'application lorsque celle-ci sera déployée en Haïti.

De plus, il faudra également fournir une documentation complète et détaillée dans le but d'aider ces futurs développeurs lorsqu'ils devront reprendre l'application.

4.3 Structure des données hors-ligne

Pour que l'utilisateur puisse accéder aux données en étant hors-ligne, il faut bien évidemment que celles-ci soient stockées sur l'appareil. Pour ce, faire il a fallu trouver une solution peu gourmande en stockage, accessible rapidement en lecture et permettant la modification des données. Deux solutions ont alors été envisagées.

JSON Stocker les réponses au format JSON telles qu'envoyées par le serveur dans le cache du navigateur. L'avantage de cette solution est qu'il est très rapide de récupérer les données à afficher sur la page lors du chargement de celle-ci, ces données étant stockées telles qu'utilisée par le navigateur. Cette solution

est également très facile à mettre en place. Si notre choix c'était porté sur l'accès statique aux données décrit dans la section 4.1, cette solution aurait été suffisante. Mais comme il faut pouvoir modifier les données, cette solution n'est pas la plus adaptée : il faudrait mettre le JSON en mémoire, le modifier et ensuite réécrire tout le nouveau JSON dans le cache à chaque modification de données.

DB Utiliser un système de gestion des données intégré au navigateur. Cette option a pour avantage d'avoir un système complet de gestion de données. Les données enregistrées dans le navigateur, il est très facile de lancer des requêtes (consulter, ajouter, modifier, supprimer) sur celles-ci. Par contre, l'inconvénient est que lors de la synchronisation des différentes tables dans le navigateur, l'appareil doit fournir un travail supplémentaire.

La solution retenue est la seconde. En effet, elle se montre plus optimal dans la gestion des données stockées localement (consulter, ajouter, modifier, supprimer).

Chapitre 5

Implémentation

5.1 Application de base

Dans cette section, un résumé de l'application HaïtiWater créée précédemment sera présenté. Pour de plus amples informations sur les raisons des choix technologiques suivants et les détails de leur implémentation, le document [3] est en consultation libre.

Backend

L'application HaïtiWater est une application web propulsée par le langage Python. C'est ce langage qui a été choisi pour le backend grâce à sa popularité et sa facilité d'apprentissage. Il existe énormément de bibliothèques logicielles et de documentations sur internet.

Vu la taille du projet, un Framework a dû être utilisé, l'organisation et la structure du framework permettant une productivité plus élevée qu'un développement sans Framework et facilitant la maintenance grâce une bonne organisation du code source. Le choix s'est porté sur Django principalement pour ses outils de gestion de base de données particulièrement efficaces pour les données géographiques.

Le Systeme de Gestion de Base de Donn[Pleaseinsert\PrerenderUnicode{ÃI}intopreamble]es (SGBD) utilisé pour l'application est PostgreSQL. Ce dernier a été choisi pour deux raisons :

- PostgreSQL est le système recommandé par Django, celui-ci s'adaptant le mieux à l'Object relational mapping. C'est Django qui va se charger d'effectuer la connexion avec la base de données et qui va se charger d'y envoyer toutes les requêtes,
- dispose de l'extension PostGis permettant de traiter facilement les données géographiques.

Django utilise une architecture modulaire facilite l'évolution et la maintenance de l'application. La figure 5.1 représente les relations entre les différents modules tels que présentés dans le mémoire de la première version de l'application [3].

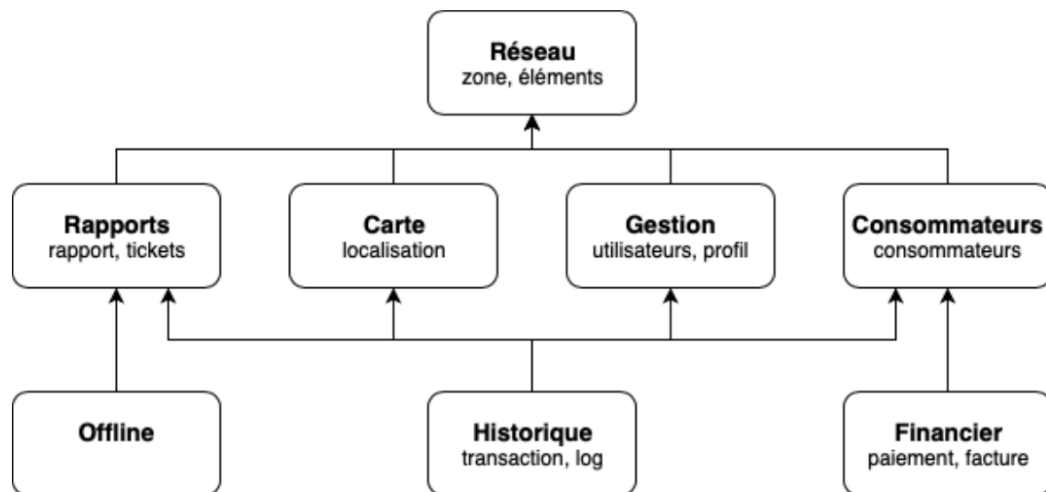


FIGURE 5.1 – Relations entre les modules de l'application HaïtiWater

Le serveur possède également une API qui lie l'Object relational mapping (ORM) de Django et les requêtes du client pour récupérer des données dans la base de données. Les requêtes concernant une page à afficher et celles concernant les données sont donc séparées. Cela facilite la récupération des données par les différentes bibliothèques du frontend et, en cas de création d'une autre application, permet de continuer à récupérer les informations sur le serveur.

Frontend

Pour le design du frontend, la librairie bootstrap a été utilisée. Elle permet l'agencement de et la personnalisation de l'interface graphique au moyen de blocs. La raison de son utilisation est qu'il s'agit d'une bibliothèque très connue et utilisée, donc facile à apprendre.

Pour l'affichage des données, deux autres bibliothèques ont été utilisées. La première, "datatable.js", permet l'affichage des données sous forme de table, leur recherche et le lancement de leur modification. La seconde, "chart.JS", permet l'affichage des données sous forme de graphiques. Toutes les données sont récupérées par les blibliothèques via l'API du serveur.

Client

L'application utilise la mise en cache afin d'éviter les téléchargements répétitifs des mêmes fichiers. Pour gérer cette mise en cache, le service worker a été utilisé comme Middleware entre le client et le serveur afin d'intercepter les requêtes pour les mettre en cache.

5.2 Choix technologiques

Une étude a été faite pour savoir qu'elle était l'option la plus adaptée pour que l'application puisse être accessible lorsque l'utilisateur n'a plus de connexion au serveur. Plusieurs options ont alors été envisagées.

Application android

Cette option est la plus facile à implémenter pour gérer le fonctionnement hors-ligne de l'application. Dans ce cas, l'application Android cohabiterait avec l'application web existante. Cela aurait permis aux gestionnaires de fontaines d'accéder à une version limitée de l'application sur leur téléphone ou tablette quand ils sont sur le terrain.

L'avantage principale de créer une application dédiée aux mobiles est la facilité de développement des fonctionnalités hors-ligne. De plus, l'intégration de certaines fonctions natives de l'appareil mobile (camera, localisation...) dans l'application permet une utilisation plus simple de l'outil. Un premier inconvénient est la création totale d'une nouvelle application. Un second inconvénient est la maintenance de deux codes sources différents.

Le problème de cette solution est qu'elle ne fonctionne que sur Android et pas sur les appareils Apple utilisant le système iOS. De plus elle oblige le maintien et l'évolution deux codes sources différents, ce qui demanderait plus de temps ou deux équipes de développement.

Xamarin

Xamarin [5] est une plateforme open source qui permet de créer des applications modernes et performantes pour Android, iOS et Windows grâce au ".NET". L'avantage de cette solution est qu'il suffit d'utiliser la partie API existante du serveur pour créer une application qui serait disponible sur tout type d'appareils.

Le défaut de Xamarin est qu'il faut un temps d'apprentissage relativement long avant utilisation. Cette technologie n'est pas ultra répandue et il sera donc plus difficile pour les développeurs de Haïti de faire maintenir et évoluer le projet. Même s'il s'agit d'une solution multi-plateforme, il faut tout de même écrire autant d'interfaces différentes que de plateformes cibles (Android, iOS et Windows).

React-native

Cette technologie permet d'écrire des applications qui seront disponibles à la fois pour Android et iOS. Il est même possible de dériver facilement du React-native pour en faire une application web React. Cela permettrait de ne pas trop diversifier les codes sources et ainsi de faciliter leur maintenance.

La difficulté réside dans le fait que, même si React-native permet de faire du multi-plateforme, il est nécessaire de connaître partiellement le langage et les API natives de la plateforme cible (Android et iOS). Bien que cette solution commence à être de plus en plus utilisée, elle reste plus complexe à maintenir. De plus, cette solution oblige tout de même à continuer le maintien de deux applications différentes : l'application web et l'application React-native.

Progressive web-app

Les PWA sont de simples applications web utilisant les nouvelles capacités des navigateurs modernes. Grâce aux service worker, l'application fonctionne lorsque la connexion au serveur n'est pas disponible. Grâce à l'app-manifest, celle-ci devient "installable" sur l'appareil de l'utilisateur. L'installation et les mises à jour se font sans intervention de l'utilisateur. Installée, une PWA peut tourner en plein écran pour faire croire qu'il s'agit d'une application native. Ce n'est bien sûr par réellement le cas, l'appareil va ouvrir une page du navigateur en plein écran pour donner l'illusion.

De plus aucune installation n'est nécessaire de la part de l'utilisateur et les mises à jour de l'application sont faites de manière transparente. L'inconvénient de cette solution est que la gestion du mode hors-ligne est plus complexe, on ne peut pas profiter des capacités de l'appareil mobile aussi bien qu'avec une application native et cette technologie étant assez récente, il n'y a pas énormément de ressources d'aide en ligne.

5.2.1 Choix définitif

La solution choisie pour faire évoluer le projet est la PWA. Les raisons de ce choix sont :

- Le maintien d'un seul code source contrairement aux autres solutions citées,
- La possibilité de réutiliser entièrement et d'améliorer le code source existant,
- La diminution de la charge du serveur et l'amélioration de l'expérience utilisateur permises par l'usage du service worker et l'app-manifest.

5.3 Client

Toute cette partie décrira le fonctionnement du service worker. Un service worker est placé en middleware entre une application web et le navigateur. C'est lui qui va permettre une utilisation de l'application déconnectée du serveur. Il va intercepter toutes les requêtes faites par le navigateur et va effectuer différentes actions en fonction. C'est via ce service worker que s'établit l'accès aux nouvelles APIs des navigateurs [7].

5.3.1 Stratégie de synchronisation des pages

Toutes les pages de l'application sont synchronisées à l'aide du service worker et sont stockées dans un des caches du navigateur. Les fichiers nécessaires à l'affichage des pages suivent différentes stratégies de synchronisation en fonction de leur usage.

Fichiers statiques

Les feuilles de styles et les fichiers JavaScript ne sont synchronisés qu'une seule fois, lors de l'installation du service worker. Ils seront mis à jour à chaque fois que le service worker est remplacé dans le navigateur. Cela se produit lorsqu'une nouvelle version de celui-ci est disponible.

Si les développeurs veulent forcer la récupération de nouveaux fichiers statiques, il leur suffit de changer la variable "cacheVersion" dans le code du service worker. Si jamais certains de ces fichiers n'ont pu être téléchargés lors de l'installation, ceux-ci seront automatiquement mis en cache lorsque le navigateur de l'utilisateur les téléchargera pour afficher les pages demandées.

Pages personnalisées

Ces pages ne contiennent pas de contenu qui changent dynamiquement. Par contre, elles contiennent du contenu "personnalisé" en fonction de l'utilisateur qui est connecté à l'application (son nom, sa zone, les résumés de sa zone...). Le côté personnalisé de ces pages étant créé par Django, il faut pouvoir mettre à jour ces pages régulièrement au cas où les données à afficher sur celles-ci auraient changé.

C'est la raison pour laquelle on va utiliser le mode de synchronisation "Stale While Revalidate". Le principe est qu'à chaque fois qu'une requête est lancée pour récupérer ces pages, le service worker répond d'abord avec la page qui est stockée dans le cache. Il va, dans le même temps, transmettre la requête au serveur afin de pouvoir mettre à jour l'ancienne page stockée dans le cache. Ces pages sont mises en cache lorsque l'utilisateur se connecte pour la première fois à l'application et sont supprimées lorsqu'il se déconnecte. Les pages concernées par ce mode de synchronisation sont les suivantes :

- /accueil
- /offline
- /aide
- /profil/editer
- /consommateur
- /reseau

Pages fixes

Ces pages ne contiennent aucun élément personnalisé en fonction de l'utilisateur ou aucun élément qui doit être mis à jour. Le côté dynamique de ces pages est entièrement géré par les bibliothèques "Datatable.js" et "Graph.js" qui vont récupérer les données à afficher directement via l'API du serveur.

Toutes ces pages sont également chargées lorsque l'utilisateur se connecte pour la première fois et elles se suppriment également lorsque celui-ci se déconnecte. La suppression de ces pages lorsque l'utilisateur se déconnecte n'est pas réellement nécessaire mais, pour une raison de sécurité, ces pages sont tout de même supprimées à la déconnexion. Les pages concernées par ce type de synchronisation sont les suivantes :

- /carte
- /gestion
- /historique
- /rapport
- /finances

5.3.2 Stratégie de synchronisation de la DB

Pour gérer l'accès aux données hors-ligne, j'ai utilisé l'API de bas niveau indexedDB. Il s'agit d'un système de gestion de bases de données orienté objet basé sur JavaScript où sont stockés des objets indexés avec une clé.

Les opérations effectuées par indexedDB sont réalisées de manière asynchrone et transactionnelle. Cela permet de ne pas bloquer le fonctionnement de l'application lorsque des données sont en cours de chargement [6]. Pour faciliter l'usage d'indexedDB qui n'est pas forcément simple à prendre en main, j'ai utilisé la bibliothèque "Dexie.js" [2]. Celle-ci offre des fonctionnalités qui facilitent le développement :

- Une meilleure gestion des erreurs,
- Une meilleure gestion des requêtes,
- Un code simplifié et plus court.

Récupération des données

Lors du premier chargement de l'application, le service worker va envoyer des requêtes à l'API du serveur afin de récupérer toutes les données nécessaires à l'affichage des différentes tables. Toutes ces requêtes sont lancées de manière asynchrone afin de ne pas bloquer l'application.

Au-delà de la première synchronisation automatique, l'utilisateur peut décider lui-même via l'interface de mettre à jour toutes les tables ou juste certaines d'entre elles. La figure ?? montre la récupération des données des consommateurs d'eau potable lors de la connexion de l'utilisateur et la manière dont le client récupère ces données en étant hors-ligne.

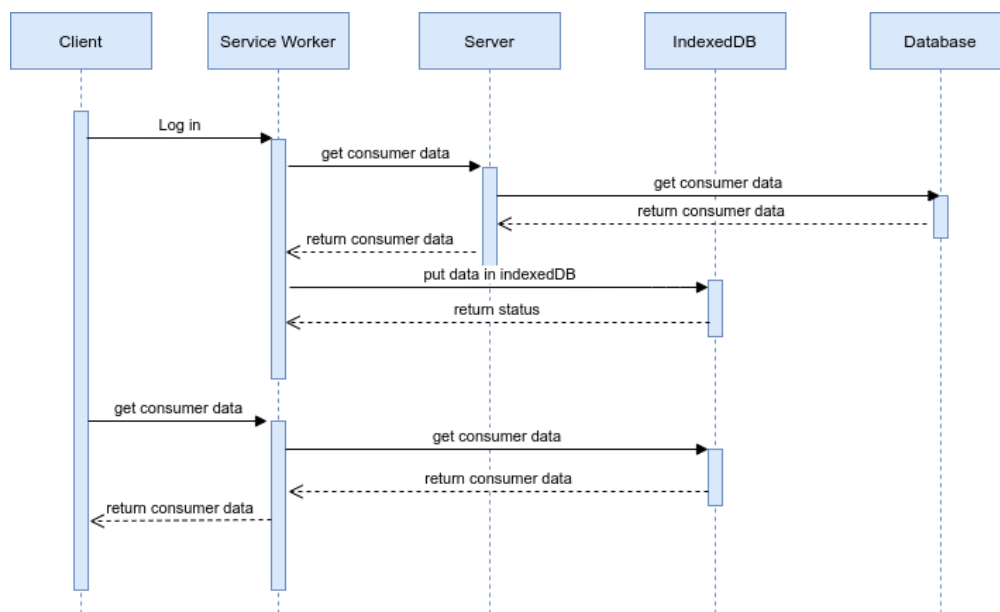


FIGURE 5.2 – Diagramme de séquence du chargement des données

Pour éviter d'avoir un dédoublement des données tout en conservant le principe de la base de données non-relationnelles, les données liées qui doivent s'afficher dans la même table sont toutes regroupées dans la même collection. Cela permet de conserver une grande vitesse de lecture des données et de ne pas ralentir le fonctionnement de l'application.

Pour accélérer le processus, toutes les requêtes pour récupérer les données sont envoyées simultanément. Dès qu'une réponse est reçue, le service worker commence à ajouter les données dans l'indexedDB à l'aide de la bibliothèque "Dexie.js" [2]. Pour ce faire, le service worker parcourt simplement tout le fichier JSON reçu de l'API et ajoute toutes les données présentes dans la collection adéquate.

Envoi des données

Pour gérer l'envoi des données, la première solution envisagée a été d'utiliser la nouvelle API "background-sync". Quand le service worker détecte qu'une requête réseau a échoué, il peut l'enregistrer comme un événement "sync" qui sera délivré quand le navigateur a récupéré la connexion.

L'avantage de cette solution est que le service worker peut envoyer les données même si la page a été fermée, ce qui permet d'éviter que les données restent indéfiniment non envoyées. Malheureusement, cette API n'est pour l'instant prise en charge que par certains navigateurs sur PC et Android mais n'est pas du tout prise en charge par iOS. Il a donc été décidé de laisser cette solution de côté.

Du coup pour pallier ce problème, j'ai implémenté une solution qui est compatible avec tous les navigateurs. Lorsque l'utilisateur essaie d'ajouter, modifier ou supprimer des données, celles-ci sont dans un premier temps stockées dans l'indexedDB avant d'être envoyées. En cas de perte de connexion au serveur, cela permet de ne pas perdre les données que l'utilisateur vient d'encoder. Les données enregistrées dans l'indexedDB, le service worker va automatiquement essayer de les envoyer. En cas de réussite, les données sont supprimées de l'indexedDB et en cas d'échec, les données restent stockées afin de pouvoir être envoyées plus tard. A chaque fois que l'utilisateur tentera de récupérer des données sur le serveur, le service worker va d'abord essayer d'envoyer toutes les données en attente avant de récupérer les données du serveur. Cela permet d'éviter d'avoir des données non envoyées qui restent en attente.

Si l'envoi des données réussit, le serveur répond avec les nouvelles données mises à jour. Lorsque le service worker reçoit cette réponse, il va mettre à jour les données qui sont dans le navigateur afin que l'utilisateur puisse voir directement les changements qui ont été effectués même s'il utilise le mode hors-ligne de l'application ???. La gestion des incohérences sera décrite plus loin dans la section 5.5.

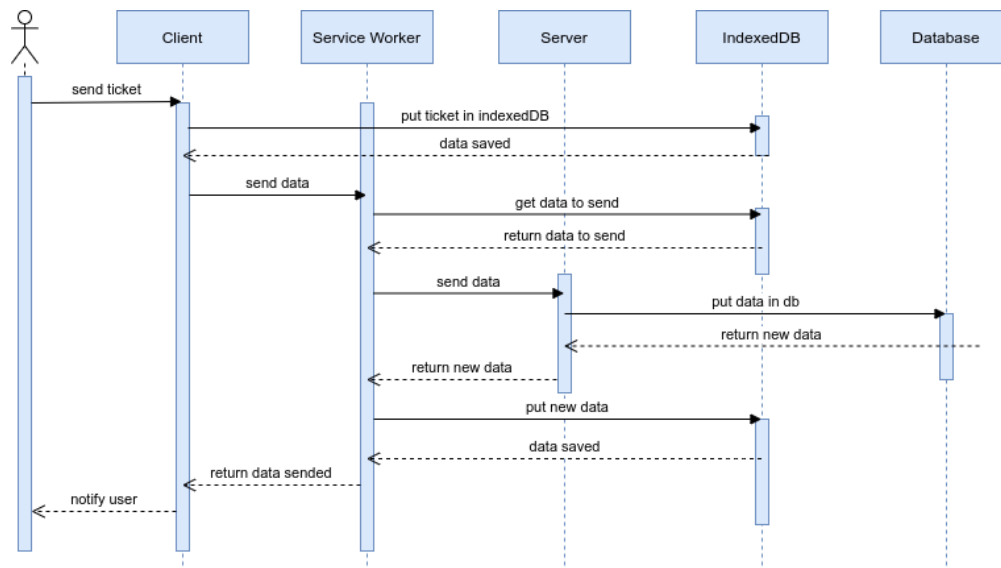


FIGURE 5.3 – Diagramme de séquence de l'envoi de données

5.3.3 Gestion des changements d'utilisateurs

Afin de conserver une sécurité maximale, à chaque déconnexion d'un utilisateur, toutes les données et les fichiers (hors fichiers statiques) sont supprimés du navigateur. Il aurait été possible de ne pas supprimer ces données et de gérer des fonctionnalités multi-utilisateurs mais cela aurait impliqué de demander à l'utilisateur d'effectuer des démarches supplémentaires pour supprimer ces données.

De plus, à l'heure actuelle, chaque utilisateur possède son propre appareil ; il n'est donc pas nécessaire de gérer le multi-utilisateur au détriment de la sécurité. En effet, en laissant les données dans le navigateur après la déconnexion d'un utilisateur, les données restent accessible via l'inspection du navigateur. Si une personne mal intentionnée utilisait l'appareil après un utilisateur, elle pourrait récupérer toutes les données stockées.

5.3.4 Système de notifications

Le système de notification, comme indiqué dans le tableau 4.1, ne faisait pas partie des fonctionnalités à implémenter pour l'instant. Toutefois, un système local de notifications a tout de même été mis en place pour notifier l'utilisateur. Ces notification concernent :

- La mise en attente d'envoi,
- La réussite ou l'échec d'envoi,
- La réussite ou échec de synchronisation,
- Autres informations.

5.4 Interface utilisateur

L'interface utilisateur existante a dû subir quelques modifications afin d'accueillir le mode hors-ligne. La nouvelle application permet en effet deux modes différents d'utilisations.

5.4.1 Changement de mode

Afin de passer d'un mode d'utilisation à l'autre, un bouton switch a été ajouté en haut de l'interface, à côté de la cloche de notification. Quand ce bouton affiche un rond vert, cela signifie que l'application est en mode en ligne de l'application. Lorsque le rond est noir, cela signifie que l'application est en mode hors-ligne.

A côté de ce bouton switch a été ajouté un autre bouton permettant de mettre à jour toutes les données présentes dans le navigateur, cela afin de ne pas obliger l'utilisateur à mettre à jour toutes les tables une par une. À droite de ce bouton, se trouvent la date et l'heure des données les plus anciennes présentes dans le navigateur. Lorsque l'utilisateur se connecte à l'application, la phrase "pas encore de données" s'affiche, aucune donnée n'étant encore chargée. Cela permet à l'utilisateur de savoir à quel point les données du navigateur sont à jour.



FIGURE 5.4 – Nouveaux boutons

Mode en ligne

Il s'agit du mode par défaut lorsque l'utilisateur se connecte à l'application. Il faut du temps à toutes les données soient chargées dans le navigateur afin que l'application puisse fonctionner en mode hors-ligne. Programmer ce mode par défaut permet donc à l'utilisateur d'utiliser l'application bien que toutes les données ne soient pas encore chargées. Dans ce mode, les différentes pages vont récupérer les données à afficher directement dans la base de données grâce à l'API et pas dans l'indexedDB du navigateur.

Ce mode est très utile s'il est nécessaire d'avoir accès aux dernières données disponibles et quand la connexion le permet. Pour de plus amples informations sur ce mode de fonctionnement vous pouvez consulter le mémoire de la première version de l'application [3].

Mode hors-ligne

Ce mode permet d'afficher les données qui sont présentes localement dans l'indexedDB du navigateur. C'est dans ce mode que l'utilisateur pourra utiliser l'application lorsque la connexion au serveur n'est pas disponible. En passant en mode hors-ligne, la couleur du bandeau de l'application passe du bleu au rouge, signifiant que l'utilisateur est bien dans le mode hors-ligne, voir ???. Les données ne sont donc plus récupérées sur le serveur.

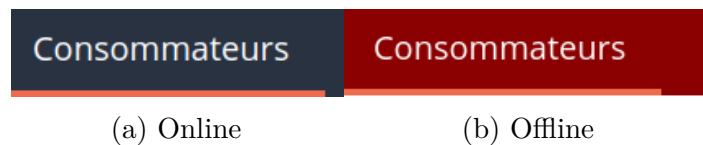


FIGURE 5.5 – Modes d'utilisation

Dans ce mode, lorsque des requêtes n'ont pu être envoyées, les données concernées apparaîtront en mauve dans le tableau pour que l'utilisateur sache que des modifications sont en attente. Quand les données ont été envoyées et que le serveur a répondu, les données hors-ligne modifiées sont mises à jour localement pour refléter les changements effectués sur le serveur. Elles n'apparaissent plus en mauve dans les tables, voir 5.6.


11	Agenordera	SAraj	Femme	Voie Cardijn	2478844888	6	Fontaine Siwèl	-100	CAEPA Passe Catabois	 
12	Abella	Chantale	Femme	La Baie	36522698	6	Fontaine Wout Simetyè (La Baie)	-195	CAEPA Poste Métier	 

FIGURE 5.6 – Données en attente de synchronisation

5.4.2 Module à synchroniser

Il s'agit d'un nouveau module qui a été ajouté pour les besoins du mode hors-ligne. Dans ce module, l'utilisateur retrouve un tableau qui reprend toutes les données qui n'ont pas été envoyées vers le serveur. Dans ce tableau, il peut retrouver différentes informations, voir 5.7.

Actions effectuées					
Cliquez sur une ligne pour afficher les détails ! Validez ou refusez les changements avec les boutons d'action.					
Date	Table	Type	Id	Status	Actions
30/05/2021, 15:36	Rapport mensuel	Ajouter	?	En attente	<input checked="" type="checkbox"/> <input type="checkbox"/>
ID élément réseau : 77 Status : Actif Jours de fonctionnement : 18 Heures de fonctionnement : 13 Mois : mai M³ : 1241 Prix M³ : 31231€ Total : 1312€					
30/05/2021, 15:35	Historique	Accepter	3805	En attente	<input checked="" type="checkbox"/> <input type="checkbox"/>
30/05/2021, 15:14	Consommateur	Editer	11	En attente	<input checked="" type="checkbox"/> <input type="checkbox"/>

FIGURE 5.7 – Module "A synchroniser"

L'utilisateur peut choisir de supprimer des données du tableau, auquel cas il ne pourra plus essayer de les envoyer, celles-ci étant supprimées de l'indexedDB. Une alternative consiste à envoyer une donnée en particulier ou toutes en même temps grâce au bouton prévu situé dans l'en-tête du tableau.

5.5 Serveur

Dans cette partie, je vais décrire le fonctionnement du serveur. Je ferai d'abord un résumé sur le fonctionnement de base du serveur et j'expliquerai ensuite les changements apportés au serveur afin de récupérer les données nécessaires à l'affichage hors-ligne.

Requêtes

Une requête est une demande que le navigateur envoie au serveur afin de récupérer différents types de fichiers ou de données. Le serveur permet de gérer trois types de requêtes différentes.

Fichiers statiques Le serveur répond à ces requêtes en envoyant tous les fichiers de style et Javascript.

Documents Le serveur renvoie les pages à l'utilisateur l'HTML à afficher. Ce sont les documents qui vont importer les fichiers statiques.

Données Le serveur renvoie toutes les données demandées. Pour ce faire l'Object-relational mapping est utilisé. Django se charge d'effectuer la connexion avec la base de données et de lui renvoyer toutes les requêtes. Cela permet d'interagir avec les données sans avoir à toucher au SQL.

Les fichiers statiques et les documents permettent au navigateur d'afficher toute l'interface graphique. Chaque module du serveur permet de desservir une page de l'application. Ces pages peuvent avoir des éléments personnalisés en fonction de l'utilisateur connecté. Les données sont téléchargées après le chargement de la page.

N'étant pas l'objet de mon mémoire, le fonctionnement global du serveur n'a pas été modifié.

API

Le module API a subi quelques modifications afin que le service worker puisse télécharger toutes les données dont il a besoin lorsque la connexion au serveur n'est pas disponible. La version de base du serveur ne permettait que le téléchargement d'une partie des données, celle à afficher directement par la page. Par exemple, lorsque l'application doit afficher la page "Consommateurs", l'API ne répond que les dix éléments devant être affichés sur la page et une nouvelle requête est nécessaire à chaque fois que des données différentes sont souhaitées (la page 2 du tableau par exemple).

Pour résoudre ce problème, j'ai implémenté des fonctions supplémentaires qui permettent de récupérer en une seule requête toutes les données nécessaires à l'affichage d'une table. Cela permet au service worker de récupérer toutes les données dont il a besoin afin de les mettre dans l'indexedDB lors de son installation. Toutefois, ces nouvelles fonctionnalités nécessitent que l'utilisateur soit connecté.

De plus, afin d'aider le mode hors-ligne à être plus précis dans l'affichage des données, lorsqu'une requête d'ajout, de modification ou de suppression de données, le serveur ne va pas répondre qu'avec un status 200 (signifiant que les changements ont bien été pris en compte), mais va également renvoyer la donnée modifiée au format JSON afin que le service worker puisse mettre à jour directement les données

localement. Le JSON est un format de données textuelles qui permet de représenter l'information sous forme de texte.

Une autre difficulté se manifeste quand les utilisateurs ne sont plus connectés au serveur ; il est possible qu'ils modifient la même donnée sans concertation

Pour l'instant le travail des gestionnaires de fontaines doit être validé par un gestionnaire de zone via le module "Historique" décrit au point 2.2.7. Le serveur accepte les données comme elles viennent et les ajoute dans la liste des données en attente de traitement. Si des incohérences apparaissent, le gestionnaire responsable contacte les gestionnaires concernés par l'incohérence et la corrige en approuvant ou en rejetant les modifications apportées. Pour l'instant, la donnée qui apparaît sur le serveur est celle de l'utilisateur qui a envoyé ces données en dernier.

Chapitre 6

Validation

La validation a pour objectif de tester le bon fonctionnement et la qualité du produit "fini". Pour effectuer cette validation, deux types de test ont été utilisés : d'une part, des tests automatisés afin de pouvoir tester le fonctionnement des différentes fonctions implémentées et, d'autre part, des tests de validation avec des utilisateurs réels afin de tester le fonctionnement global de l'application ainsi que son intuitivité.

6.1 Vérifications automatiques

Pour tester l'application de façon automatique, j'ai utilisé des tests unitaires. Ces tests ont pour objectif de vérifier le bon fonctionnement d'une partie précise d'une application (unité ou module). Ils assurent qu'une méthode qui peut être utilisée par un utilisateur fonctionne de la manière prévue. Dans le cas présent, toutes les interactions entre l'utilisation et l'utilisateur se font à l'aide soit de requêtes envoyées directement au serveur, soit de requêtes faites au service worker. J'ai donc décidé de compléter les tests unitaires qui étaient déjà présents pour la partie serveur et de créer une batterie de tests unitaires pour le service worker.

Pour de plus amples informations le mémoire en lien avec la création de l'application est en consultation libre [3]. Les tests que j'ai ajoutés au niveau des requêtes au serveur couvrent les méthodes de récupération des données utilisées

par le service worker. Je vérifie que les données récupérées sont uniquement celles auxquelles l'utilisateur a accès et que celles-ci sont complètes.

En ce qui concerne les tests sur le service worker, ceux-ci couvrent toutes les fonctions de récupération et d'envoi de données de et vers la base de données ainsi que la gestion de toute page mise en cache pour leur affichage en mode hors-ligne.

6.2 Vérifications utilisateurs réels

. Lors de cette phase, l'application a été présentée à des utilisateurs (certains ayant utilisés la première version, d'autres non) afin qu'ils puissent fournir le feedback nécessaire à son évolution. Trois aspects ont été testés : la compréhension des différents modes, la gestion des données au cas de perte de connexion au serveur et les modifications de l'interface. Le but est d'identifier les problèmes présents dans l'application et de réfléchir à leur solution. Dans cette optique, avoir des avis externes sur l'application permet de l'améliorer drastiquement.

Les tests se sont déroulés en présentiel avec les participants ou via Microsoft Teams en fonction de la disponibilité des personnes et des mesures sanitaires en vigueur. Un groupe de dix personnes venant d'horizons différents et ayant des compétences diverses et variées ont testé l'application.

Méthodologie

La méthodologie qui a été utilisée pour faire passer ces tests est restée la même que pour le mémoire précédent [3]. Celle-ci a été adaptée aux nouvelles fonctionnalités développées lors de ce mémoire afin que ce test reste cohérent avec le travail réalisé. Le fait d'utiliser le même type de test permet de voir concrètement comment évolue le niveau de satisfaction des utilisateurs après cette deuxième année de développement.

Les séances de validation se sont déroulées en partie physiquement et en partie à distance via le logiciel Microsoft Teams, ces logiciels permettant de faire de la transmission vocale et du partage d'écran. L'étudiante de Haïti m'a également aidé

à faire passer les tests aux différents utilisateurs présents là-bas, cela afin de me faciliter la tâche à cause du décalage horaire.

Les testeurs ont d'abord reçu une brève introduction à l'application et sur le but de la séance de validation. Je leur ai ensuite remis un à trois scénarios à réaliser durant la séance. Ces scénarios contiennent une brève introduction au contexte et une liste de tâches à compléter. Après la lecture du scénario, je répondais aux questions éventuelles. Ensuite, l'utilisateur est laissé seul pour faire sa tâche et est chronométré. S'il avait des questions pendant le déroulement de la tâche, je lui donnais une indication pour l'aider.

Après la réalisation de ces tâches, le testeur a reçu une série de 21 questions. Ces questions sont regroupées par thème.

Tâches Ces questions permettent de savoir si les tâches à accomplir étaient suffisamment claires.

Fonctionnalités Ces questions permettent de savoir si l'interface de l'application permet une utilisation simple des fonctions implémentées.

Usabilité Ces questions permettent d'avoir le ressenti de l'utilisateur et permet de récolter des indices quant à une possible amélioration de l'application au niveau l'usabilité.

Esthétique Ces questions permettent de savoir si le design de l'application plaît aux utilisateurs.

Pour répondre aux différentes questions, j'ai utilisé l'échelle de Likert. L'échelle de Likert est une échelle de mesure qui comprend dans ce cas cinq degrés de réponse. Elle est souvent utilisée dans les questionnaires. Elle permet de jauger le niveau d'accord ou de désaccord d'une affirmation [1].

Résultats obtenus

Le graphe 6.1 reprend les résultats des 21 questions posées aux utilisateurs pour évaluer leur niveau de satisfaction sur les thèmes cités à la section 6.2. Les résultats sont dans l'ensemble très positifs. Cela montre que les utilisateurs apprécient cette nouvelle version de l'application et la trouvent simple à utiliser.



FIGURE 6.1 – Données de validation concernant le test de l'application

Il y a cependant un point qui n'a pas mis tout le monde d'accord ; à l'affirmation "Je trouve le fait d'avoir deux modes d'utilisation inutilement complexe", le panel de réponse est hétérogène. Cette solution ne fait pas l'unanimité et une évolution de la gestion des modes reste nécessaire.

Modifications apportées

Grâce aux différents commentaires reçus des utilisateurs, certains changements mineurs ont été effectués. Le premier est l'ajout d'un mode d'emploi sur la page d'accueil pour expliquer le fonctionnement du mode hors-ligne. Le second est l'ajout de notification lorsque les données sont en cours d'envoi. Les autres modifications qui peuvent être apportées seront décrites dans la section 7.3.

Chapitre 7

Améliorations futures

Malgré les deux années de travail effectuées par les trois anciens mémorants [3] et moi-même, le projet est loin d'être terminé. Même si l'application peut commencer à être déployée dès maintenant, comme tout projet de développement de grande ampleur, HaïtiWater devra être suivie afin d'être mis à jour et maintenu pour un bon fonctionnement au quotidien.

7.1 Suite du projet

S'atteler à la finition de l'application HaïtiWater sera la priorité des futures équipes de développement. En l'état actuel, l'application peut déjà servir aux acteurs locaux afin de travailler sur le terrain. Cependant, la base de l'application peut toujours être optimisée pour être encore plus sûre et réactive. Comme évoqué par mes prédécesseurs [3], une des tâches importantes pour le futur reste l'automatisation des rapports en agrégeant les données et en développant une meilleure visualisation. Une autre amélioration évoquée par mes prédécesseurs est le développement d'un serveur mail et d'un serveur SMS dédié de sorte que le serveur de l'application ne soit pas surchargé.

L'application HaïtiWater est en cours de déploiement sur un serveur haïtien et devrait être disponible dans les semaines à venir. Cette migration devrait améliorer le temps de chargement de l'application pour les acteurs locaux. Le serveur qu'ils utiliseront devrait d'ailleurs être plus performant que celui utilisé en Belgique. Pour

l'instant, il s'agit d'un serveur dual-core de 2 GHz avec 2 Go de mémoire vive. Les temps de chargement devraient donc être drastiquement réduits grâce à un serveur plus puissant.

Une dernière amélioration serait l'optimisation de l'API afin que la récupération des consommateurs soient plus rapide.

Actuellement l'application prend du temps à être complètement chargée pour une utilisation hors-ligne. Ce délai empêche l'utilisation du mode hors-ligne avec Mozilla Firefox pour les utilisateurs ayant beaucoup de consommateurs, le problème étant que ce navigateur stoppe le service worker bien avant d'avoir reçu la réponse du serveur.

7.2 Défis rencontrés

La difficulté la plus importante lors de ce projet a été la méconnaissance de la technologie utilisée et le fait qu'il y a encore peu d'aide sur internet. De plus, les service worker sont encore en cours de développement ; l'ensemble de leurs fonctionnalités ne sont pas supportées par tous les navigateurs. Il a donc fallu trouver des solutions pour que l'application puisse rester fonctionnelle même si l'utilisateur emploie un navigateur qui ne prend pas en charge les service worker. Une période conséquente d'apprentissage autonome avant de se lancer dans le projet a été nécessaire.

Une deuxième difficulté était mon manque d'expérience. En effet, je ne m'étais jamais engagé dans un projet d'une telle envergure. Heureusement, les anciens mémorants étaient assez disponibles et m'ont aidé à résoudre les problèmes rencontrés.

Le dernière difficulté rencontrée a été une motivation en dent de scie. Comme tout le monde, mes conditions de vie (isolement social) et de travail (manque d'objectifs à court terme) ont été bousculées par les mesures sanitaires faisant suite à la pandémie du COVID-19.

7.3 Propositions

L'application HaïtiWater peut encore évoluer de nombreuses manières. Une demande a été faite pour intégrer un module Citizen Science afin que les consommateurs du réseau de distribution d'eau potable puissent signaler eux-mêmes les différents problèmes présents sur ce réseau ainsi que consulter leurs factures et consommations.

En l'état l'application a été optimisée grâce à la mise en cache de tous les fichiers nécessaires à l'affichage des pages et à la copie locale de la base de données. Cependant, une amélioration possible serait l'utilisation d'un Framework frontend (Vue, React ou Angular) afin d'améliorer la réactivité globale de l'interface, ces Framework s'intégrant parfaitement aux PWA.

La gestion des données stockées localement pourrait être améliorée afin de ne plus proposer deux modes différents de fonctionnement de l'application mais un seul mode dans lequel toutes les données seraient synchronisées automatiquement au fur et à mesure que l'utilisateur adresse ses requêtes au serveur.

Une autre évolution possible serait d'ajouter un système de notifications basé sur le serveur plutôt que localement. Pour l'instant, ce système de notifications n'est pas du tout synchronisé avec le serveur et ne sert qu'à l'affichage de certaines informations stockées dans le navigateur. Un service de messagerie interne à l'application pourrait également aider les utilisateurs à communiquer entre eux. Ce système de notifications mis en place, la gestion des incohérences de données décrite dans la section 5.5 pourra être améliorée afin d'envoyer des avertissements aux deux utilisateurs qui ont modifié les mêmes données hors-ligne.

Enfin, les modules existants pourraient être améliorés afin de proposer différentes façons d'interagir avec les données. par exemple, il serait intéressant de pouvoir exporter la base de données sous forme de fichier excel ou d'intégrer d'avantages d'outils graphiques rendant les informations plus visuelles.

Chapitre 8

Conclusion

Grâce à la réalisation de mon mémoire, j'ai pu m'inscrire dans un projet d'une grande envergure, mettre en pratique les compétences acquises au cours de mes études et acquérir une expérience non négligeable dans le développement d'une application web de grande ampleur.

Lors de la phase de validation avec des utilisateurs réels, l'application HaïtiWater et son nouveau mode hors-ligne ont suscité un intérêt certain chez les testeurs. Leurs retours étaient plutôt positifs. J'ai bon espoir que l'application finira par être utilisée par les acteurs locaux de manière régulière, voire quotidienne, et qu'elle continuera à évoluer.

Avec plus de temps et une plus grande latitude dans les tâches à accomplir, j'aurais pu mettre en place les améliorations précitées à la section 7.3. De plus, je regrette de ne pas avoir pu procéder à la validation plus tôt, ce qui m'a limité dans le champ des améliorations. Je n'ai malheureusement pu apporter que des améliorations mineures.

Cette expérience d'apprentissage m'aura permis de mieux définir dans quel domaine de l'informatique je désire travailler après l'obtention de mon diplôme. De plus, cela m'a permis de définir quelles technologies je souhaiterai investiguer et maîtriser dans un futur plus ou moins proche.

Bibliographie

- [1] Stephane Contrepolis. Echelle de likert, 2021. Consulté pour la dernière fois le 8 juin 2021.
- [2] dexie.org. dexie.js, 2021. Consulté pour la dernière fois le 24 mai 2021.
- [3] Celine Deknop, Sebastien Sterbelle et Adrien Hallet. Haïtiwater : Développement d’une application web pour gérer la distribution de l’eau en haïti, Juin 2019.
- [4] Laurent Granger. Moscow, 2021. Consulté pour la dernière fois le 6 juin 2021.
- [5] Microsoft. Xamarin, 2021. Consulté pour la dernière fois le 20 mai 2021.
- [6] Mozilla.org. Indexeddb, 2021. Consulté pour la dernière fois le 24 mai 2021.
- [7] Mozilla.org. Service worker, 2021. Consulté pour la dernière fois le 23 mai 2021.

Annexe A

Diagramme de Gantt

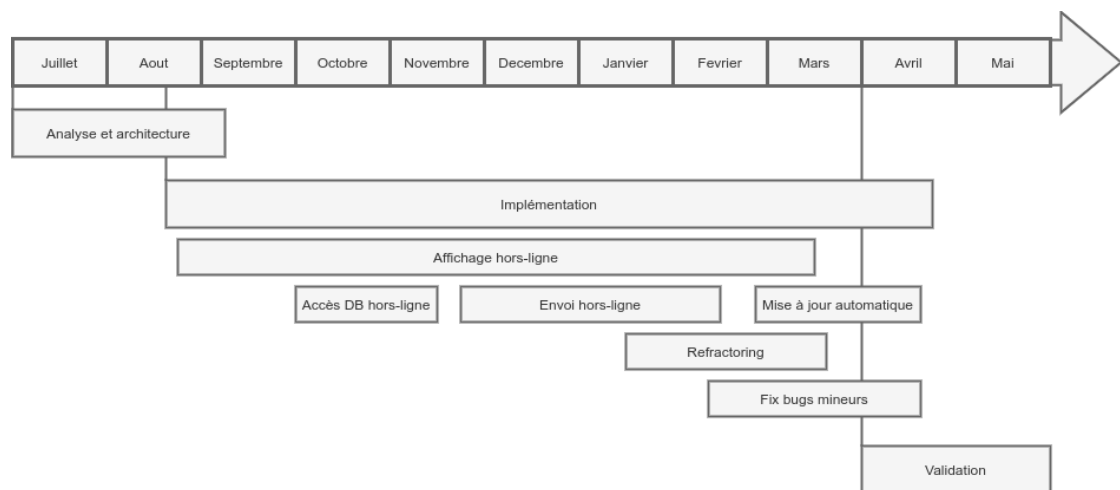


FIGURE A.1 – Diagramme de Gantt

Analyse et architecture La première phase de mon mémoire est une phase d'analyse permettant la détermination des technologies à utiliser selon les différentes fonction à implémenter. C'est dans cette phase qu'il y a eu beaucoup de discussions avec l'étudiante venue de Haïti.

Implémentation La deuxième phase de mon mémoire, l'implémentation, est la plus conséquente ; c'est ici que toutes les décisions prises durant la phase d'analyse se sont mises en place. Cette phase se divise en plusieurs phases qui représentent les différentes fonctionnalités produites durant le mémoire.

Validation La phase de validation n'intervient que tard dans le déroulement de mon mémoire. Pour cause, des facteurs tels que la méconnaissance de la technologie à utiliser et le temps de déploiement de la nouvelle version de l'application sur les serveurs haïtiens ont retardé sa mise en oeuvre.

Annexe B

Documents de validation

B.1 Questionnaire

Voici les 21 questions posées aux utilisateurs pendant la validation. Elles sont triées par thème.

Tâches

- Je comprends les tâches demandées.
- Je comprends l'utilité des tâches demandées dans le contexte de la gestion de l'eau.
- Je comprends l'utilité de l'application.

Fonctionnalités

- Je comprends l'utilités des contrôles de l'interface.
- Je peux réaliser les tâches facilement.
- Je peux réaliser les tâches rapidement.
- Je comprends le fonctionnement du mode hors-ligne.
- Le module "A synchroniser" est compréhensible.
- Je comprends comment envoyer des données en attente.
- Le système de notification est compréhensible.

Usabilité

- Je pense que j'aimerais utiliser l'application fréquemment.
- Je trouve le fait d'avoir 2 modes d'utilisation (hors-ligne et en ligne) inutilement complexe.
- L'application est facile à utiliser.
- J'ai trouvé les différentes fonctionnalités du mode hors-ligne bien intégrées.
- Je pense que la plupart des utilisateurs apprendraient à utiliser l'application rapidement.
- Je suis confiant en utilisant l'application.
- J'ai besoin d'apprendre beaucoup de choses avant de pouvoir utiliser l'application.
- J'ai besoin de l'aide d'une personne qualifiée pour utiliser ce système.

Esthétique

- J'aime les couleurs de l'application (pour le mode en-ligne et hors-ligne).
- L'agencement des différents boutons me convient.
- L'esthétique générale de l'application me satisfait.

B.2 Scenario 1 - Gestion hors-ligne

Rôle Administrateur principale

Objectif Créer un nouveau gestionnaire et lui assigner une zone.

Prérequis La zone de l'artibonite dans la DB.

Contexte Vous êtes Claude, membre de l'ONG Protos. L'application HaïtiWater est déjà utilisée dans plusieurs départements d'Haïti et le département de l'Artibonite, zone nouvellement créée, vous souhaitez créer un nouveau gestionnaire de zone et l'assigner à celle-ci. En tant que responsable de l'application, vous devez permettre au responsable de la gestion de l'eau en Artibonite de se connecter à l'application et de gérer son réseau. Vous devez donc créer un nouveau gestionnaire et lui assigner la zone d'Artibonite. Lorsque vous voulez encoder les informations, le réseau disparaît soudainement.

Informations nécessaires **Utilisateur :** Protos

Mot de passe : Protos

Tâche 1

- Connectez-vous à l'application avec votre compte de gestionnaire : Protos.
- Pendant que les données charges (indiqué en haut de l'écran), voyager dans l'application.
- Vous êtes dans la zone de l'Artibonite pour rencontrer le nouveau gestionnaire et l'encoder.
- Vous allez dans la partie gestion et là la connexion au serveur disparaît.
- Passez l'application en mode hors-ligne.
- Encodex le nouveau gestionnaire et assignez-lui sa zone (vous pouvez entrer vos propres données ou des données fictives).
- Vérifiez dans la partie "A synchroniser" que les données ont bien été enregistrées pour être envoyées ultérieurement.

Tâche 2

- Le réseau est revenu.
- Allez dans la partie à synchroniser.
- Vérifiez que les données encodées précédemment sont correctes.
- Envoyez les données vers le serveur.
- Retournez dans la partie dans la partie gestion et vérifier que vos changements ont bien été pris en compte.

B.3 Scenario 2

Rôle

Objectif

Prérequis

Contexte

Informations nécessaires

Mes infos

Tâche 1

-

Tâche 2

-

B.4 Scenario 3

Rôle

Objectif

Prérequis

Contexte

Informations nécessaires

Mes infos

Tâche 1

-

Tâche 2

-

B.5 Scenario 4

Rôle

Objectif

Prérequis

Contexte

Informations nécessaires

Mes infos

Tâche 1

-

Tâche 2

-

B.6 Scenario 5

Rôle

Objectif

Prérequis

Contexte

Informations nécessaires

Mes infos

Tâche 1

-

Tâche 2

-

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl