Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Behind Food

Amélioration de l'application mobile sur la face cachée des aliments industriels

Rapport

Semestre 5
2021-2022

Étudiant : Grégory Geinoz
gregory.geinoz@edu.hefr.ch

Professeur : Pascal Bruegger
Conseiller/Mandant : Samuel Fringeli

Hes·so // FRIBOURG FREIBURG
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Version : 1.0
Fribourg, 12.10.2021

Table des matières

1 Introduction

Behind food est une application mobile utilisée dans le cadre d'une exposition sur le développement durable. Elle permet aux visiteurs d'explorer les faces cachées de différents aliments du quotidien au moyen d'un parcours de divers thèmes reliés à ces aliments. Ce parcours donne accès à des images, à des vidéos et à des textes pour illustrer les caractéristiques des aliments concernés. Ces éléments sont mis à jour par l'équipe qui gère l'exposition, au moyen d'une interface backend, et sont accessibles depuis l'application grâce à une API. Dans la version actuelle de l'application, c'est une WebView qui est chargée et qui affiche les images et vidéos au fur et à mesure du parcours de l'utilisateur dans la structure, mais les médias ne sont pas sauvegardés dans le stockage local de l'application, ce qui rend impossible l'utilisation de celle-ci hors-ligne. Comme l'exposition a pour objectif de fonctionner entièrement hors-ligne, il serait nécessaire de faire en sorte que les données affichées soient téléchargées localement, avec un système permettant d'actualiser les dernières modifications effectuées sur le backend par l'utilisateur de l'application.

1.1 Contexte

L'application de base [1] est une application web qui utilise la bibliothèque ZircleUII [2], une interface utilisateur simple mais intelligente avec une navigation zoomable intégrée à travers des cercles, affichée dans une application iOS avec un WebView de SwiftUI. Tout le contenu affiché, y compris le texte, les images et les vidéos, est obtenu à partir d'une API créée par le mandant.

1.2 Objectifs

Le projet est constitué d'objectifs principaux et secondaires. Les objectifs principaux sont à réaliser en priorité.

1.2.1 Objectifs principaux

1. **Application multiplateforme**
   Titre auto-explicatif, l'idée est d'adapter l'application iOS de base en une application multiplateforme. La technologie utilisée pour ce projet est documentée dans la section Analyse de ce document entre Cordova, Flutter ou Xamarin.

2. **Compatibilité hors-ligne**
   L'application doit devenir compatible hors-ligne. Cela signifie qu'elle est capable de vérifier la connexion de l'appareil et de s'adapter selon cette dernière. Elle doit être capable de stocker son contenu, de le récupérer si l'appareil est hors-ligne, et de mettre à jour le contenu si l'API a été mise à jour quand l'appareil est en ligne.

1.2.2 Objectifs secondaires

1. **Mises à jour**
   Ajouter les applications Android et iOS respectivement sur le Play Store et l'App Store et gérer les futures éventuelles mises à jour de ces applications mobiles.

1.3 Outils de collaboration

Le code et les documents officiels de ce projet sont postés sur le GitLab de l'école (autorisations à demander) et les réunions entre les différents acteurs du projet se font soit en présentiel au bureau de M.Bruegger (D20.17) soit avec Microsoft Teams.

1.4 Structure du rapport

Le projet est organisé selon un modèle hybride cascade et agile avec la structure suivante :

1. Analyse
2. Conception
3. Implémentations
4. Tests
5. Conclusion
6. Documentation

L'analyse permet de trouver les directions avec lesquelles les implémentations sont codées. Puisqu'il s'agit d'une amélioration d'une application déjà existante, la conception n'est qu'accessoire et ajoute simplement les objectifs à la conception existante. Les implémentations sont en fait des sprints référençant chaque objectif du projet. Chaque objectif est implémenté, testé et présenté au professeur et au mandant afin d'être confirmé pour pouvoir passer au sprint suivant. Une fois ces sprints réalisés, la phase de test confirme les implémentations ou permet de les corriger. La phase de documentation se déroule sur l'entier du projet, en commençant par ce document qui est écrit au fur et à mesure de l'avancement du projet.

2 Analyse

2.1 Les solutions multiplateformes

Il s'agit pour ce premier objectif de convertir l'application de base iOS en une version cross-plateforme en utilisant une des tehcnologies décrites dans les chapitres ci-dessous. L'application de base affiche simplement l'application web dans une WebView du WebKit de iOS [3]. Afin de s'orienter au mieux pour le deuxième objectif, il faut que cette technologie supporte non seulement les WebViews mais également l'API IndexedDB [4] qui est nécessaire pour la persistance des données au sein d'un navigateur internet.
Les dernières versions de Chrome et de Safari pour toutes plateformes confondues sont respectivement la version 94 [5] et 15 [6], sorties toutes deux très récemment.

2.1.1 Xamarin

Xamarin est une plateforme open-source qui permet de créer des applications mobiles sur iOS, Android et Windows Phone. Le code partagé de l'application s'écrit en C# [7]. Un des frameworks de Xamarin est Xamarin.Forms [8], qui fournit les WebView [9]. Lors de la rédaction de ce document et durant une bonne partie du semestre 5, Xamarin est un sujet majeur du cours "Développement Mobile Cross-Plateforme".

2.1.2 Cordova

Apache Cordova est un framework de développement mobile open-source. Il permet de créer des applications multi-plateformes en utilisant les technologies web standards actuelles HTML5, CSS3 et JavaScript. Cordova utilise de base des WebViews afin d'afficher tout l'interface graphique des applications, ce framework est donc un bon choix pour s'implifier l'utilisation des WebViews pour les applications multiplateformes. De plus, une documentation officielle explique l'utilisation de IndexedDB avec Cordova [10], ce qui est rassurant pour simplifier l'implémentation du second objectif.

2.1.3 Flutter

Flutter est un kit de développement logiciel (SDK) open-source créé par Google. Il permet de créer des applications multiplateformes sur Android, iOS, mais aussi Windows, Linux, Mac et le web à partir d'une seule base de code, le Dart. Le package webview_flutter permet d'importer les WebViews dans une application Flutter. Malheureusement, Flutter bloque IndexedDB et le seul moyen de contourner cette restriction est d'utiliser un plugin qui est en fait un simple wrapper Flutter pour IndexedDB, mais il semble que Flutter ne permette pas de mettre en cache des gros fichiers [11].

2.1.4 Conclusion

Flutter ne semble pas une bonne solution pour les objectifs de ce projet. Cordova peut être intéressant car il s'agit déjà en quelque sorte d'une application web pour mobile qui fait tourner une autre application web. Xamarin est également une bonne solution même si aucune documentation officielle ne mentionne l'utilisation de IndexedDB.

2.1.5 Choix final

TBD

2.2 Les solutions hors-ligne

Afin de rendre l'application existante utilisable sans connexion, il faut que les données récupérées depuis l'API du mandant puissent persister d'une manière ou d'une autre. Pour ce faire, deux possibilités se présentent :

2.2.1 Convertir l'application web en application mobile native

Convertir entièrement l'application web avec une solution multiplateforme vue plus haut, afin d'accéder facilement au stockage de l'appareil mobile utilisé.

2.2.2 Convertir l'application web en application web multiplateforme

Améliorer l'application web actuelle afin qu'elle puisse accéder au cache des navigateurs sur lesquels elle tourne pour y enregistrer les données qu'elle utilise, afin de les récupérer facilement sans connexion dans le cas où le réseau tomberait ou qu'aucune connexion n'est disponible.

2.2.3 Conclusion

Tout dépend donc à quel endroit les données sont stockées.
La première solution permet d'utiliser le stockage interne de l'appareil mobile et de sortir du champ d'application d'une WebView, mais demande une bien plus grande quantité de codage et de prise en main, quelque soit la technologie cross-plateforme choisie.
La seconde solution ne demande qu'une adaptation cross-plateforme de l'application mobile actuelle utilisant simplement une WebView afin d'afficher l'application web convertie en application "web-mobile" lancée dans une WebView, mais demande d'utiliser les caches des navigateurs car il n'est pas possible de sortir du champ d'application de ces derniers.
L'idée de la seconde solution est de faire comme cette application web d'exemple qui, lors de la première visite avec une connexion, stocke toutes les vidéos (quelques Mo par vidéos, un peu comme l'application web actuelle) ainsi que les fichiers statiques comme le CSS et le JavaScript dans une IndexedDB. Grâce à ce processus, mettre son appareil en mode avion (ou dans un scénario plus réaliste, perdre sa connexion) n'empêche pas l'utilisation du site. Les vidéos s'affichent et peuvent toujours être lues sans aucun problème. Comme mentionné plus haut, les dernières versions de Safari et Chrome sont les versions 15 et 94, IndexedDB est parfaitement supporté sur ces deux versions récentes qui se mettent pour la grande majorité des appareils automatiquement à jour [12].
En contrôlant la connexion de l'appareil dès le démarrage de l'application [13] [14], il est possible dans le cas où il y a une connexion de vider toute la IndexedDB et de télécharger tous les fichiers de l'API afin de mettre leurs versions à jour dans la IndexedDB. Dans le cas où il n'y a pas de connexion, l'application va directement chercher dans sa IndexedDB les fichiers afin de créer l'application web sans avoir à télécharger quoi que ce soit du net. **Ceci inclut que la toute première utilisation de l'application se fasse avec une connexion internet.** Mais puisque l'objectif secondaire est de déployer cette application sur les stores, les utilisateurs désireux de l'acquérir n'auront pas le choix d'avoir une connexion internet, ce qui rend cette précondition plutôt triviale.

2.2.4 Choix final

TBD

7 Figures et Tables

7.1 Liste des figures

7.2 Liste des tables

8 Bibliographie

[1] Samuel Fringeli. Behind food. https://github.com/samuel-fringeli/behind-food, 2021. [En ligne; Page disponible le 11-octobre-2021].

[2] Juan Martin. Zircleui. https://zircleui.github.io/docs/, 2019. [En ligne; Page disponible le 11-octobre-2021].

[3] Apple Inc. ios wkwebview. https://developer.apple.com/documentation/webkit/wkwebview, 2020. [En ligne; Page disponible le 11-octobre-2021].

[4] W3C. Indexed database api 3.0. https://www.w3.org/TR/IndexedDB/, 2021. [En ligne; Page disponible le 11-octobre-2021].

[5] WhatIsMyBrowser.com. What's the latest version of chrome? https://www.whatismybrowser.com/guides/the-latest-version/chrome, 2021. [En ligne; Page disponible le 11-octobre-2021].

[6] WhatIsMyBrowser.com. What's the latest version of safari? https://www.whatismybrowser.com/guides/the-latest-version/safari, 2021. [En ligne; Page disponible le 11-octobre-2021].

[7] Microsoft. Qu'est-ce que xamarin ? https://docs.microsoft.com/fr-fr/xamarin/get-started/what-is-xamarin, 2021. [En ligne; Page disponible le 11-octobre-2021].

[8] Microsoft. Qu'est-ce que xamarin.forms ? https://docs.microsoft.com/fr-fr/xamarin/get-started/what-is-xamarin-forms, 2021. [En ligne; Page disponible le 11-octobre-2021].

[9] Microsoft. Xamarin.forms webview. https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/user-interface/webview?tabs=windows, 2021. [En ligne; Page disponible le 11-octobre-2021].

[10] Apache. Store data - indexeddb. https://cordova.apache.org/docs/en/10.x/cordova/storage/storage.html#indexeddb, 2021. [En ligne; Page disponible le 11-octobre-2021].

[11] Hossam Tarek. How do i import the dart indexeddb library in flutter? https://stackoverflow.com/questions/67490597/how-do-i-import-the-dart-indexeddb-library-in-flutter, 2021. [En ligne; Page disponible le 11-octobre-2021].

[12] Alexis Deveria. Can i use... indexeddb. https://caniuse.com/indexeddb, 2021. [En ligne; Page disponible le 11-octobre-2021].

[13] Microsoft. Xamarin.essentials: Connectivity. https://docs.microsoft.com/en-us/xamarin/essentials/connectivity?tabs=android, 2019. [En ligne; Page disponible le 11-octobre-2021].

[14] Apache. cordova-plugin-network-information. https://cordova.apache.org/docs/en/10.x/reference/cordova-plugin-network-information/, 2021. [En ligne; Page disponible le 11-octobre-2021].