Documentation PlaneLookUp

Présentation du projet

But

Ce projet a pour but de créer une application mobile basé sur le framework Cordova, élargir nos connaissances en JavaScript et pour avoir la capacité de recréer une application dans un but professionnelle.

Cahier des charges

- Utiliser le framework Cordova
- Utiliser au minimum 1 plugin asynchrone
- Utiliser une librairie graphique
- Utiliser la session et/ou le local storage
- Utiliser des librairies externes
- Interagir avec une api externe
- Fournir une documentation technique

Description de l'application

L'application se nomme PlaneLookUp. Elle a pour but principal de visualiser les avions qui se situe au-dessus d'une ville. Elle permet de pouvoir sauvegarder en PDF ou partager les informations. Elle a pour fonction secondaire de stocker des mots de passe de manière sécurisé. Pour finir, elle peut scanner des QR-Code et donne la potentialité de copier le texte dans le presse-papier du téléphone. L'application gère l'accès à la localisation qui est utilisé dès l'ouverture de l'application pour se situer et observer les avions au-dessus de sois. Elle gère aussi la connexion à internet.

Plugins utilisés

Pour la localisation, j'utilise le plugin `cordova-plugin-geolocation`

Pour la gestion du réseau, c'est le plugin `cordova-plugin-network-information`

Pour gérer les QR-Code, j'utilise le plugin `cordova-plugin-qrscanner`

J'utilise le plugin `cordova.plugins.diagnostic` et `cordova-open-native-settings` pour gérer si l'application a les autorisations et le cas échéant, savoir quelles sont les autorisations, savoir lesquelles manque.

Pour la sécurité des mots de passe, c'est le plugin `cordova-plugin-fingerprint-aio` qui s'occupe de l'utilisation de l'empreinte.

La sauvegarde du PDF sur le téléphone se fait grâce au plugin `cordova-plugin-file`

`cordova-plugin-x-socialsharing` est le plugin qui s'occupe du partage.

Pour copier le texte sur le presse-papier, le plugin utilisé est `cordova-plugin-Flipboard-x`

Pour changer l'écran de chargement de Cordova, j'utilise `cordova-plugin-splashscreen`

Pour certaines actions, le téléphone vibre grâce au plugin `cordova-plugin-vibration`

Le plugin `es6-promise-plugin` est utilisé par le plugin `cordova-plugin-x-socialsharin`

Api utilisées

Open Sky

Pour pouvoir accéder à certaine information des avions, j'utilise l'API opensky-network.org. Pour ce faire, j'utilise cette demande :

`https://opensky-network.org/api/states/all?lamin=wwww&lomin=xxxx&lamax=yyyy&lomax=zzzz`

- `opensky-network.org` : Le serveur contacté
- `/api` : Dans la partie api
- `/states` : Dans les pays
- `/all` : On prend tous les avions
- `?lamin=wwww&lomin=xxxx&lamax=yyyy&lomax=zzzz` : On prend en paramètre des cordonnées pour récupérer que les avions qui sont dedans
  - `lamin=wwww` : Latitude minimum
  - `lomin=xxxx` : Longitude minimum
  - `lamax=yyyy` : Latitude maximum
  - `lomax=zzzz` : Longitude maximum

Open Cage Data

Pour avoir la localisation des villes et pouvoir récupérer les cordonnées, j'utilise l'API de opencagedata.com. Pour ce faire, j'utilise deux types de demande :

`https://api.opencagedata.com/geocode/v1/json?q=ville&key=key`

`https://api.opencagedata.com/geocode/v1/json?q=latitude%2Clongitude&key=key`

- `api.opencagedata.com` : Le serveur contacté
- `/geocode` : Dans la partie géolocalisation
- `/v1` : La première version
- `/json` : Donne une réponse en JSON
- `?q=ville&key=key`
  - `q=ville` : Le nom d'une ville pour avoir les coordonnés maximum et minimum de la ville
  - `key=key` : La clef utilisée pour s'identifier sur l'API
- `?q=latitude%2Clongitude&key=key`
  - `latitude%2Clongitude` : La latitude et la longitude pour retrouver la ville
  - `key=key` : La clef utilisée pour s'identifier sur l'API

Librairies externes utilisées

Carte

Pour pouvoir générer une carte, j'utilise la librairie `leaflet`. Sur cette carte, je limite le déplacement. Je permets à l'utilisateur de voir au-dessus de sa ville ou de la ville qu'il a choisie. Je dessine sur la carte un carré orange transparent permettant de voir la zone ou les avions sont trouvés. Et pour chaque avion trouvé, je mets un point pour voir où il se situe précisément.

PDF

Je génère les PDF grâce à `jspdf`. Pour ce faire, cette librairie utilise deux autres pour fonctionner. Ce sont les librairies `html2canvas` et `dompurify`.

Description Fonction

Code : script.js

Variable globale :

```
const ARTICLE_CHARGEMENT // L'article où tout est rangé
const PROGRESS_DEMMANDE // La bare progress
const UL_AVANCEMENT_DEMANDE // La liste pour les log
```

Ce sont les différents éléments qui permettent de faire le chargement lors de la communication avec les api.

```
const ARTICLE_INFO_AVION // L'article où tout est rangé
const DIV_AVION // On va afficher les avion et leur info
const P_INFO_AVION // On affiche le nombre d'avions
```

Ce sont les éléments qui ont un rapport avec les avions

```
const ARTICLE_INFO_VILLE // L'article où tout est rangé
const SELECT_UTILISATEUR_VILLE // Le selecteur avec les villes
const SELECT_UTILISATEUR_PAYS // Les selecteur avec les pays
```

Les éléments qui permettent de choisir une ville

```
const ARTICLE_ERREUR // L'article où tout est rangé
const P_INFO_ERREUR // On affiche l'erreur qui c'est produit
const BUTTON_ERREUR // Le button pour acceder au paramètre
const BUTTON_RECHARGEMENT // Le button pour recharger la page
```

Les éléments utilisés quand il y a une erreur qui se produit

```
const ARTICLE_LOCAL // L'article où tout est rangé
const BTN_DESA_FLASH_QR // Button pour desactiver le flash
const BTN_ACT_FLASH_QR // Button pour activer le flash
```

Les éléments du lecteur de QR-Code

```
const CHARGEMENT_LOC // L'élement pour le cahrgement
```

```
const NOTIF // L'élément pour les notification
```

```
let ville_nom, texte_qr; // Varibale qu'on utilise plusieur fois
```

Variable qu'on utilise pour le nom de la vile que l'on utilise et le texte que le lecteur QR-Code à trouver

Map et Paramètre :

```
const MAP = L.map('map').setView([0, 0], 1);  // Déclaration
```

On crée la carte et on l'affecte à la variable MAP

```
MAP.setMaxZoom(12); // Zoom maximum
MAP.setMinZoom(8); // Zoom minimum
```

On met en place un zoom maximum et minimum pour la carte

```
var ico_avion = L.icon({
        iconUrl: '../img/leaflet/ico_avion.png', // Lien image
        iconSize: [20, 20] // Taille de l'icone
});
```

On crée un marqueur spécial pour les avions

Event Listeners :

```
document.addEventListener("deviceready", onDeviceReady);
document.addEventListener("pause", onPause);
document.addEventListener("resume", onResume);
document.addEventListener("backbutton", onBackButton);
```

La mise en place des éléments de Cordova

```
document.addEventListener("online", mise_ligne);
document.addEventListener("offline", mise_horsligne);
```

La gestion des mises en ligne et la mise hors ligne

```
document.getElementById("btn_affiche_avion")
.addEventListener("click", affiche_carte);
```

La gestion du button pour afficher la carte

```
document.getElementById("partage_info")
.addEventListener("click", partage_info);
```

La gestion du button de partage des infos

```
document.getElementById('btn_qr')
.addEventListener("click", start_qr); // Démarer le scan

document.getElementById("btn_stop_qr")
.addEventListener("click", stop_qr); // Arreter le scan
```

```
document.getElementById("btn_erreur_qr")
.addEventListener("click", erreur_qr); // Acceder au paramètres
```

Pour la gestion des différents button du lecteur QR-Code

```
document.getElementById("enregistre_info")
.addEventListener("click", enregistre_pdf_info);
```

Pour enregistrer le tout dans un PDF

```
SELECT_UTILISATEUR_VILLE.addEventListener("change",
utilisateur_ville); // Changement de ville
SELECT_UTILISATEUR_PAYS.addEventListener("change",
changement_pays); // Changement de pays et chargement des villes
```

Gestion des sélecteurs de villes et pays

```
//Button pour acceder au paramètre
BUTTON_ERREUR.addEventListener("click", acces_parametre);
// Button pour recharger la page
BUTTON_RECHARGEMENT.addEventListener("click", onDeviceReady);
```

Gestion des button des erreurs

```
BTN_ACT_FLASH_QR.addEventListener("click", flash_act_qr);
BTN_DESA_FLASH_QR.addEventListener("click", flash_desa_qr);
```

Gestion de button pour activer au désactivé la lampe torche

Fonctions :

```
onDeviceReady();
```

Fonction principale, quand la page est lancé, on active le chargement de la localisation, si la connexion est activée, on vérifie en même temps s'il est enregistré la liste des villes dans le local storage. On fait, appelle, à la fonction navigator.geolocation.getCurrentPosition

```
onBackButton();
```

Fonction appeler lors de l'appui sur le button retours. En fonction de l'endroit où le se trouve sur la page, on revient sur les informations de l'avion

Si la carte est affichée, on revient sur les informations des avions

Si le lecteur QR-Code est activé, on revient sur les informations des avions

```
geolocationSuccess(position);
```

Le paramètre d'entrée est la position. On utilise cette cordonnée pour faire appelle à l'API pour avoir les coordonner de la vile

```
geolocationError();
```

On gère les erreurs du fait de ne pas avoir pu récupérer la localisation en fonction de-ci c'est la géolocalisation qui n'est pas active ou si ce sont les autorisations.

```
get(url, fonction);
```

Fonction appeler quand on veut faire appelle à une api. Les paramètres d'entrer sont l'URL de l'API et le nom de la fonction qui demande

```
statechange(event);
```

Fonction appeler quand on change d'étape pour la communication avec le serveur. Paramètre d'entrer les données de la requête

```
chargement_info(etape);
```

Pour chaque étape, on avance la barre de progression et on met le texte dans la liste des logs et pour la première, on affiche le chargement et le dernier, on cache l'élément de la requête. Paramètre d'entrée le numéro de l'étape

```
demande_ville();
```

Quand l'utilisateur fait une demande de ville spécifique, on regarde si on n'a pas déjà les infos de cette ville en local, si oui, on l'utilise sinon on fait la demande

```
utilisateur_ville();
```

On appelle la fonction pour récupérer la valeur de la ville et cacher les objets pour demander une ville

```
info_avion();
```

On récupère les cordonnés maximum et minimum de la ville et on définit une zone sur la map pour délimiter la zone de recherche et on définit les limite pour ne pas dépasser

```
affiche_avion();
```

La fonction permet d'afficher les différentes informations pour chaque avion, les donner sont récupérées dans le local strorage et je l'affiche dans la div prévue à cet effet

```
affiche_carte();
```

Fonction appeler quand on veut afficher la carte avec les différentes marquer des avions

```
enregistre_pdf_info();
```

Fonction pour enregistrer le PDF

```
partage_info();
```

Fonction appeler pour partager des informations

```
liste_avion_html();
```

Fonction pour modifier les balise de HTML pour l'adapter en texte

```
afficher_pays();
```

Permet la création des différentes options du select des pays depuis la liste des différentes villes

```
changement_pays();
```

Fonction appeler quand on change de pays pour afficher les villes de chaque pays

```
pop_up(message_etat);
```

Fonction permettant de faire afficher un message à l'utilisateur avec un popup. Paramètre d'entrée le texte à afficher

```
pop_up_qr(message_etat);
```

Fonction appeler pour faire un popup pour afficher le texte et utiliser la fonction copier. Paramètre d'entrée le texte à afficher

```
copie_texte();
```

Fonction pour mettre dans le presse-papier

```
acces_parametre();
```

Fonction appeler quand l'utilisateur clique sur le bouton de paramètre. Gère si c'est li wifi ou la localisation.

```
loc_succes(enabled);
```

Vérifie s'il a accès à la localisation et regarde si elle est activée, si oui, il envoie sur les autorisations des paramètres sinon sur les a paramètre de la localisation. Paramètre d'entrée, le statut de la localisation.

```
loc_err(error);
```

S'il n'a pas accès, il fait un message d'erreur. Paramètre d'entrée, l'erreur donné

```
mise_ligne();
```

Fonction appeler quand le téléphone est mis en ligne. Il affiche la partie si les appelle des villes est cachée

```
mise_horsligne();
```

Fonction appeler quand le téléphone est hors ligne. Il montre l'article des erreurs et cache la demande de ville

```
start_qr();
```

Fonction pour démarrer le service de QR-Code.

```
prepare_qr(err)
```

Fonction appeler pour préparer la caméra pour le QR-Code.

```
scan_qr(scan_err, texte);
```

Fonction appeler pour scanner le QR-Code. Paramètre d'entrer l'erreur du scan et le texte retourné

```
stop_qr();
```

Fonction qui arrête le scanne du QR-Code

```
flash_act_qr();
```

Fonction appeler pour activer le flash de la caméra

```
flash_desa_qr();
```

Fonction appeler pour désactiver le flash de la caméra

```
erreur_qr()
```

Ouvrir les paramètre s'il y a une erreur

Source

- https://www.npmjs.com/
- http://srv-peda.iut-acy.local/fpour/BUT/#/2