# Origine, motivations et conception de l'application…

Traveler's Tools


Go, Go, Go…

# 1 - Objectif de mon projet

La genèse de mon idée a pris sa source lors du développement du neuvième exercice nommé « Baluchon ». Pour rappel, ce projet était une application dont le but était d'offrir à un voyageur un traducteur, un convertisseur monétaire, une indication météorologique et tout cela dans une même application.

Dans mon fort intérieur il m'apparaissait évident qu'elle manquait de certains services pour remplir d'avantage sa fonction de couteau suisse du voyageur. J'en liste ici quelques-uns :

- **un assistant d'orientation** qui le moment venu se montrerait très utile lors d'une immersion dans des lieux inconnus.
- **un moyen de lister et de transmettre les coordonnées** facilement et rapidement et cela par une manière original d'encodage.
- **un outil permettant de facilité le contact avec la population** par la participation au jeux très actuel que sont les jeux de rôle, mais pas que.

Je souhaitais aussi profiter de cette dernière création pour revoir et intégrer tout un ensemble de frameworks qui m'a accompagné tout au long de cette année.

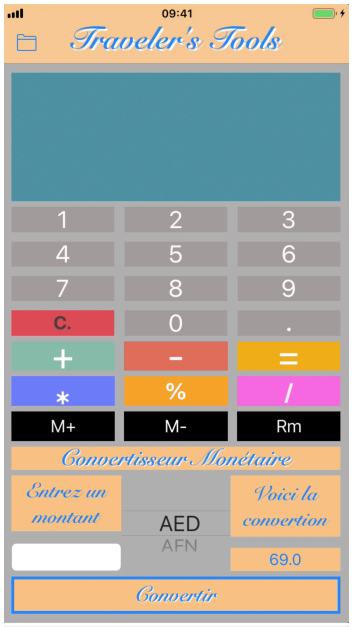
Enfin, je voulais me prouver que j'étais capable de les orchestrer sous une même application, tout en la garantissant stable et évolutive.

J'espère que cette application, accessible en open source sur Github sera installée par le plus grand nombre, afin d'obtenir de nombreux retour m'encourageant à envisager une seconde version. Celle-ci se voudra étoffe des éléments attendus et toujours dans l'esprit de proposer un outil aux globe-trotters d'où son nom « *Traveler's Tools* ».

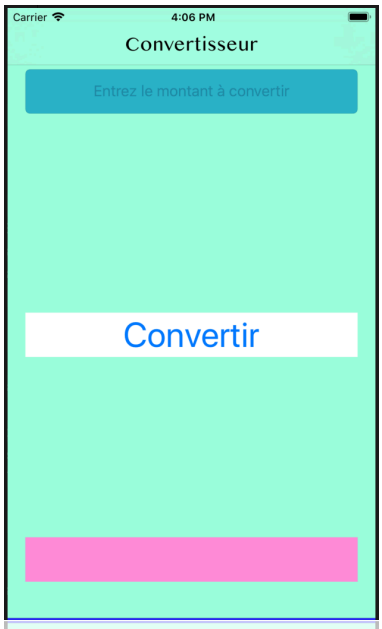
Je souhaite à tous les utilisateurs d'apprécier sans limite mon travail autant que j'ai eu de plaisir à développer cette application.

# Page d'accueil

Ma conception d'optimisation me contraint à utiliser toutes les vues. Alors dès la page d'accueil l'application devait avoir son utilité. J'y ai domicilié une calculatrice (un petit clin d'oeil a été fait à CountOnMe) dans les deux premiers tiers de la vue et dans le dernier le convertisseur monétaire, première composante du « Baluchon ».



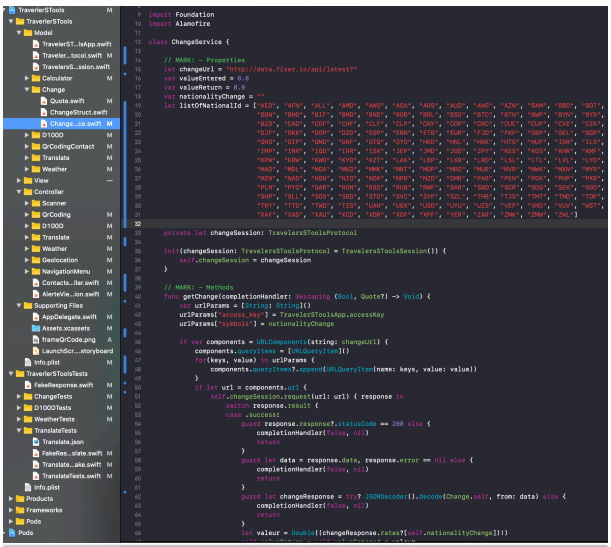

Ne nous y trompons par, il n'était pas question pour moi de faire un copier coller de mon code ni de mon graphisme. Voyez plutôt…

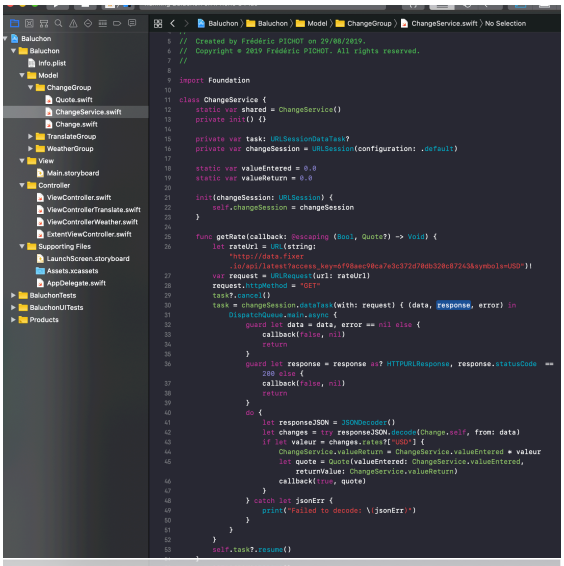



J'ai tout d'abord converti les appels réseaux des trois éléments vers « Alamofire », créant de fait un protocole et une session pour chacun, puis je les ai factorisés.

Je lui ai apporté un pickerView offrant un plus grand choix de monnaies, pour finir j'ai animé tous les boutons.

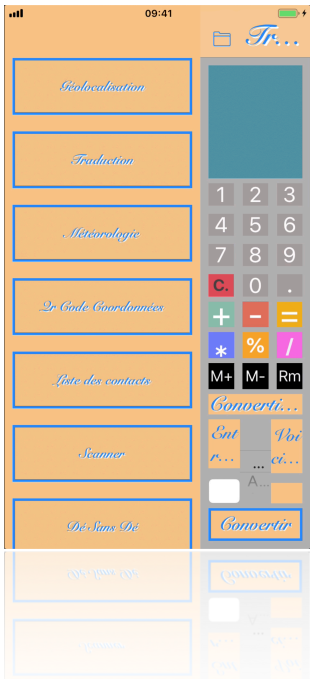
Voici les captures d'écran des deux versions.





# Container de navigation

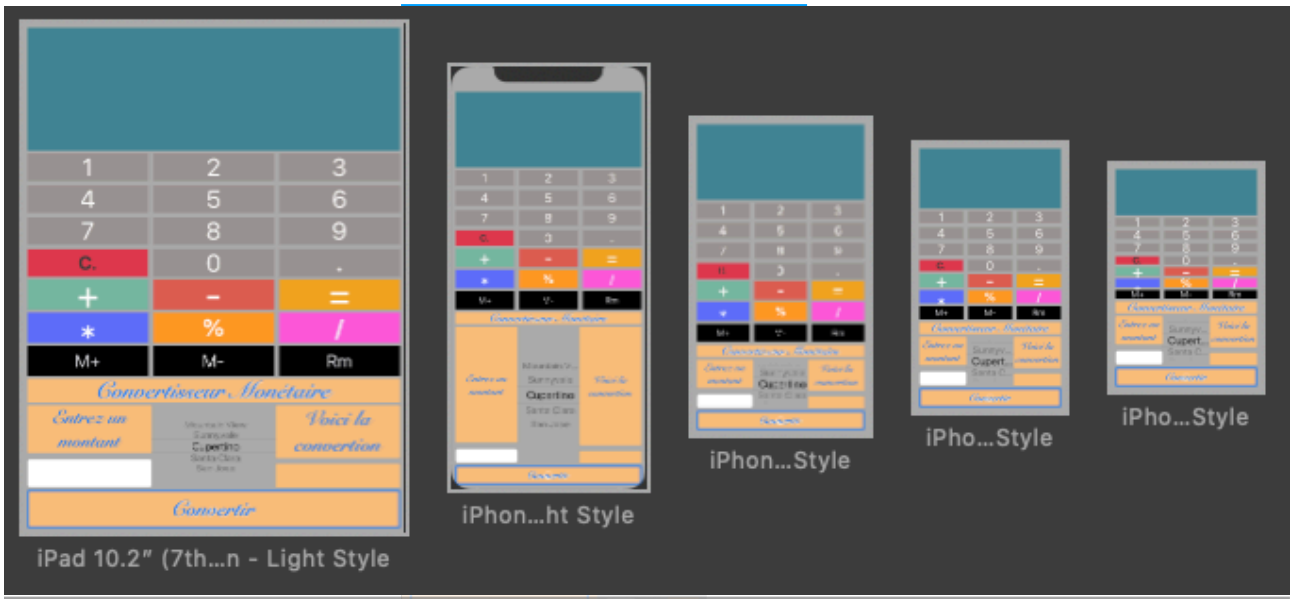
Au regard de mon ambitieux projet, la Tab-Bar me semblait sous dimensionné alors, j'ai opté pour un container sous la forme d'une « TableViewController » remplit de tableViewCell. Celle-ci ne devant apparaître que lorsque l'on clique sur ce bouton 




Chacune d'entre elle dirige l'utilisateur vers une partie de l'application.

On pourra remarquer que l'arrivé du container permet toujours l'usage des deux composantes de la vue d'accueil. (calculatrice et convertisseur). Dans mon iPhone 6, l'exercice est périlleux mais sur un écran plus grand tout redeviens visible.

Ici nous utilisons un iPhone 6, le rendu n'est pas le même sur les autres tailles d'écran (**autolayout** oblige). **Mon application est responsive**.



Géolocalisation

Le framework « MapKit »

## J'ai du modifier mon « Info.plist » pour l'autorisation

Ce qui me paraissant important dans mon application de voyage était de donner la possibilité à son utilisateur de déambuler là où il se trouve en toute liberté sans avoir à se soucier de retrouver son chemin. J'ai donc intégré une géolocalisation.

Pour plus de finesse, j'ai souhaité donner le choix du mode de déplacement, alors j'ai intégré un menu déroulant avec les principaux moyens actuels qui assombrit la vue principale lorsqu'il est déroulé.

```
case "Automobile":
    directionsRequest.transportType = .automobile
case "Walking":
    directionsRequest.transportType = .walking
case "Transit":
    directionsRequest.transportType = .transit
case "Any":
    directionsRequest.transportType = .any
```

Une fois renseigné le menu se replie, redonne toute la luminosité à la carte et libère un champs dans lequel l'utilisateur devra indiquer le lieu ou le genre d'établissement qu'il recherche. Une fois terminer il devra valider son choix en pressant sur le bouton Rechercher du clavier.

Alors, le parcours se dessine en bleu sur la carte et à chaque changement de direction un rond vert apparait.

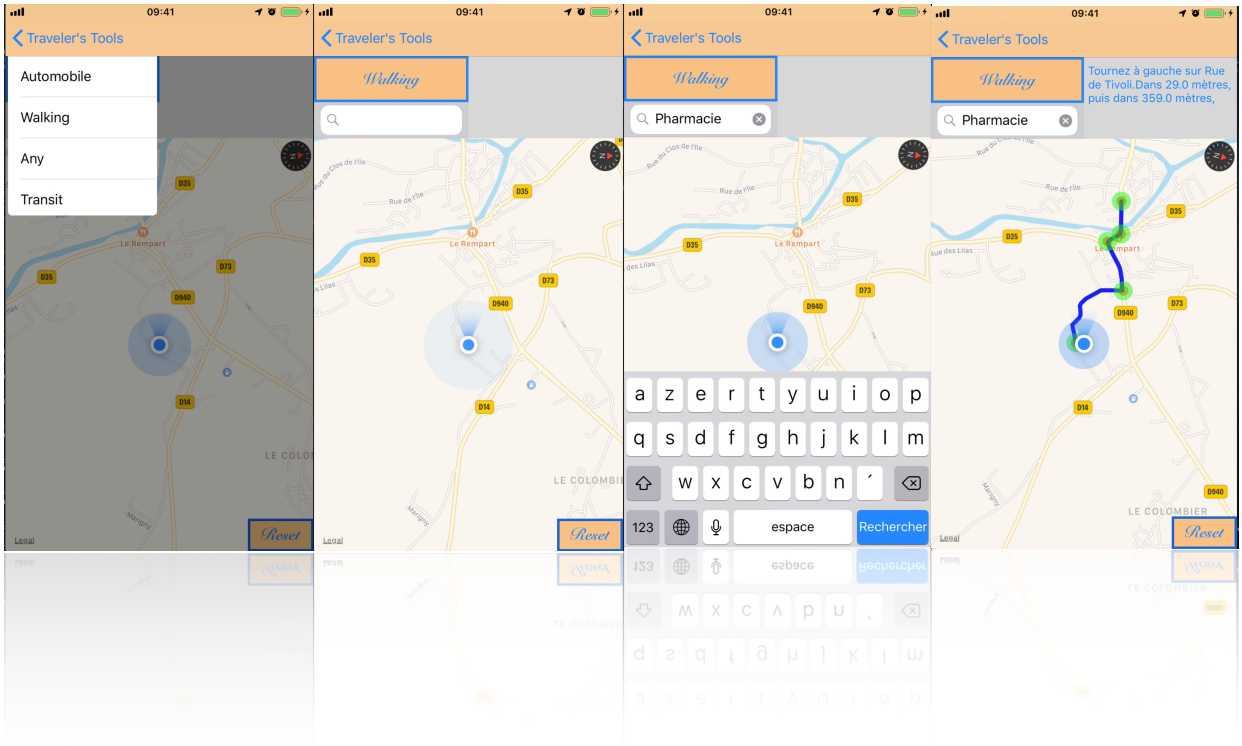
On remarquera aussi qu'en haut à droite le parcours est noté, il évoluera au fur et à mesure du déplacement.

```
let initialMessage1 = "Dans \(self.steps[0].distance) mètres,"
let initialMessage2 = "\(self.steps[0].instructions) puis dans \(self.steps[1].distance) mètres,"
let initialMessage = "\(self.steps[1].instructions)." + initialMessage1 + initialMessage2
self.directionTextField.text = initialMessage
```

et pour finir sa **lecture est audible.**

```
98  // sound of the message
99              let speechUtterance = AVSpeechUtterance(string: initialMessage)
100             self.speechSynthesizer.speak(speechUtterance)
101             self.stepCounter += 1
```

Pour toute nouvelle requête un bouton est présent.




Les deux autres composantes de « Baluchon » ont eu les mêmes transformations de leur code que le convertisseur, nous remarquons tout de même un effort sur la charte graphique.