Instagrid

Application de montage et de partage de photos

Walim ALOUI - 14 Juillet 2018

Fonctionnalités et Bonus

Fonctionnalité Corbeille

Afin d'appliquer cette fonctionnalité, j'ai rajouté un UIButton, dont l'image est une corbeille. Il a fallu ensuite connecter par un ***control-drag*** le UIButton au ***ViewController*** sous forme d'Action : ***trashButtonPushed*(_ *sender*: *UIButton*).**

```
@IBAction func trashButtonPushed(_ sender: UIButton) {
    let refreshAlert = UIAlertController(title: "Deletion", message: "All pictures will be removed. Are you sure ?", preferredStyle:
        UIAlertControllerStyle.alert)
    refreshAlert.addAction(UIAlertAction(title: "Ok", style: .default, handler: { (action: UIAlertAction!) in
        self.mainSquareView.deleteImages()
    }))
    refreshAlert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: { (action: UIAlertAction!) in
    }))
    present(refreshAlert, animated: true, completion: nil)
    manageBackGroundColor()
}
```

On observe ci-dessus la fonction permettant de supprimer l'ensemble des images présentes dans le collage. Lorsque l'utilisateur touche la corbeille, une pop-up apparaît avertissant l'utilisateur que ***toutes*** les images vont êtres supprimées. Deux choix s'offrent alors à l'utilisateur Ok, ou *Cancel*. On constate que si la réponse est Ok, une fonction de la classe ***MainSquareView***, appelée ***deleteImages()*** est alors appelée. Son action est visible ci-dessous :

```
// MARK: Deleting images managament
func deleteImages() {
    for images in imageViewsArray {
        images.image = nil
    }
    for plusButton in plusButtonsArray {
        plusButton.isHidden = false
    }
}
```

Il s'agît d'une boucle ***for*** parcourant l'ensemble du tableau ***imageViewsArray*** contenant les quatre ***imageView*** du collage. En parcourant le tableau la fonction vérifie l'état de chaque ***UIImageView*** i.e si elle contient une image ou non. Si c'est le cas, celle-ci est effacée et le bouton *plus* correspondant réapparaît afin d'offrir à nouveau la possibilité d'ajouter une photo.

Fonctionnalité Géolocalisation

La fonctionnalité de Géolocalisation est représentée par un symbole de « pointeur », en haut à gauche de l'application (mode portrait). Elle consiste en la récupération des données géographiques et leur inscription en tant que texte au sein du ***UITextField***.

Avant de commencer à écrire la fonction, il faut auparavant s'assurer de bien signaler au ***ViewController*** la présence de cette fonctionnalité en ajoutant à la suite de la ligne class ***UIViewController*** le 'délégué' de ***Core Location Manager*** ou ***CLManagerDelegate***.

En outre, dans le fichier ***info.list***, nous rajoutons les deux clefs suivantes et leurs valeurs respectives afin d'autoriser le Device à utiliser ces fonctionnalités. Ce dernier, 'poppera' un message de demande d'autorisation à l'utilisateur la première fois qu'elles sont utilisées :
Privacy - Location Always and When In Use Usage Description
Privacy - Location When In Use Usage Description

La fonction principale présente dans le ***ViewController*** et qui s'occupe de ces fonctionnalités est indiquée ci-dessous

```
// Touch The location Button to find yours
@IBAction func locationButtonTapped(_ sender: UIButton) {
    locationManager = CLLocationManager()
    locationManager.delegate = (self as CLLocationManagerDelegate)
    locationManager.desiredAccuracy = kCLLocationAccuracyBest
    locationManager.requestWhenInUseAuthorization()
    locationManager.startUpdatingLocation()
}
```

Comme l'exige Apple, nous créons ensuite une instance de ***CLLocationManager*** :
locationManager = CLLocationManager()

Puis nous assignons le *delegate* de locationManager conformément au protocole ***CLLocationManagerDelegate***
locationManager.delegate = (self as CLLocationManagerDelegate)

Ensuite nous indiquons la précision désirée, ici la plus précise (un peu gourmand en énergie et en mémoire, mais c'est plus drôle !) :
locationManager.desiredAccuracy = CLLocationAccuracyBest
Cette précision nous transmet le pays, la ville, le Code Postale, et l'adresse où l'on se trouve !

Enfin nous appelons les fonctions nécessaires la requête d'informations sur le réseau et leur téléchargement sur le Device:

locationManager.requestWhenInUseAuthorization()
locationManager.startUpdatingLocation()

Or il arrive évidemment que le réseau soit complètement indisponible (l'app reste utilisable puisqu'il est possible d'enregistrer ses collages pour plus tard), il faut donc prévoir ces situations :

```
// MARK: - LOCATION MANAGEMENT

// What happens when location fails or is disabled
func locationManager(_ manager : CLLocationManager, didFailWithError error : Error) {
    self.locationTextField.text = "Error while updating location" + error.localizedDescription
}

// Function that actually gets the location
func locationManager(_ manager : CLLocationManager, didUpdateLocations locations : [CLLocation]) {
    CLGeocoder().reverseGeocodeLocation(manager.location!) { (placemarks, error) -> Void in
        if (error != nil) {
            self.locationTextField.text = "Sorry, something wrong occured..." + error!.localizedDescription
            return
        }
        if placemarks!.count > 0 {
            let pm = placemarks![0]
            self.displayLocationInfo(pm)
        } else {
            self.locationTextField.text = "Problem with the data received from geocoder"
        }
    }
}
```

La première fonction prenant pour paramètre *didFailWithError* délivre un message d'erreur si le réseau est indisponible : *Error while updating location*

La seconde fonction consiste à effectuer un relevé de longitude et de latitude afin d'acquérir une adresse précise. Il arrive que plusieurs adresses soient disponibles en même temps d'où l'utilisation du tableau *placemarks.*
Si le tableau est vide, aucune adresse n'a été trouvée et donc un message d'erreur intermédiaire est délivré : *Sorry, something wrong occured…*

Enfin, si le tableau n'est pas vide, on choisit l'adresse présente en position 0 du tableau en appelant la fonction *diplayLocationInfo(pm)* que nous verrons par la suite. Le cas échéant, un message indiquant le problème est également délivré.

```swift
// Info accuracy and details that location asks for
func displayLocationInfo(_ placemark : CLPlacemark?) {
    if let containsPlacemark = placemark {
        locationManager.startUpdatingLocation()
        let locality = (containsPlacemark.location != nil) ? containsPlacemark.locality : ""
        let postalCode = (containsPlacemark.postalCode != nil) ? containsPlacemark.postalCode : ""
        let administrativeArea = (containsPlacemark.administrativeArea != nil ) ? containsPlacemark.administrativeArea : ""
        let country = (containsPlacemark.country != nil) ? containsPlacemark.country : ""
        self.locationTextField.text = postalCode! + " " + locality!
        self.locationTextField.text?.append("\n" + administrativeArea! + ", " + country!)
// stop update location to avoi textField to be overwritten
        locationManager.stopUpdatingLocation()
    }
}
```

La fonction *displayLocationInfo* s'assure de la bonne présence des informations demandées. Si celles-ci sont existantes, elle assigne ces informations au champ text du ***UItextField***

Bonus UITextField

UITextField est un champ de texte éditable par l'utilisateur. Situé en dessous de la grille de photos il permet à l'utilisateur de glisser ou non un court message ou une légende pour son collage. ***UITextField*** appartient à la même ***Stack View*** que le carré bleu principal ce qui permet au Model de le prendre en compte lors de l'envoi du collage final.

La propriété ***locationTextField*** de type ***UITextField*** est utilisée dans le code à différentes reprises : messages d'erreurs à afficher en cas d'échec dans la géolocalisation…

```swift
25      // The label which includes location text
26      @IBOutlet weak var locationTextField: UITextField!
27

50          super.viewDidLoad()
51          self.locationTextField.delegate = self
52          locationIcon.isHidden = false

245     func locationManager(_ manager : CLLocationManager, didFailWithError error : Error) {
246         self.locationTextField.text = "Error while updating location" + error.localizedDescription
247     }

252             if (error != nil) {
253                 self.locationTextField.text = "Sorry, something wrong occured..." +
                        error!.localizedDescription
254                 return

259             } else {
260                 self.locationTextField.text = "Problem with the data received from geocoder"
261             }

272             let country = (containsPlacemark.country != nil) ? containsPlacemark.country : ""
273             self.locationTextField.text = postalCode! + " " + locality!
274             self.locationTextField.text?.append("\n" + administrativeArea! + ", " + country!)
275     // stop update location to avoi textField to be overwritten
```