Instagrid

Le Concept :

Instagrid permet de sélectionner des photos dans votre photothèque, de les placer de différentes manières selon des positionnements définis et de partager le cadre de photos choisis par mail et/ou des réseaux sociaux par un simple glisser vers le haut en mode portrait et vers la gauche en mode paysage.

Pourquoi ne pas avoir choisie le MVC :

Instagrid ne comporte aucune logique car se sont juste des sélections d'image et nous changeons pas de vue. La vue est une entité extrêmement simple car elle n'effectue aucune logique, son rôle est simplement d'afficher ce qu'on lui demande. Le MVC est particulièrement utilisée pour les projets qui nécessitent beaucoup de logiques telles que des calculs.

Créez une application responsive :

Pour répondre au demande du cahier des charges, cette application doit s'adapter à tous les iPhones à partir du SE, en mode portrait et en mode paysage. Pour cela, j'ai commencé à travailler sur les contraintes car il faut dans 95% des cas, définir 4 contraintes afin que Xcode comprenne.

Contraintes :

- Le langage utilisé doit être Swift 4 ou supérieur

- L’application doit être disponible à partir d’iOS 11.0

- L’application est supportée par toutes les tailles d’iPhone (de l’iPhone SE à l’iPhone XS Max)

- L’application n’a pas à être disponible sur iPad

- L’application n’a pas à être disponible sur iPad

Explication de la base de l'application :

La base pour que l'application est de détecter les gestes de l'utilisateur.
On va faire cela avec la classe qui le permet comme :
UISwipeGestureRecognizer qui gère les gestes simples et c'est ce dont nous aurons besoin .
On veut que le geste soit détectable le plus tôt possible, dès que l'interface est chargée. Donc nous allons créer notre geste dans `viewDidLoad`, et appeler après que le contrôleur de vue a chargé .

```
UISwipeGestureRecognizer a besoin de trois informations pour fonctionner :
 - Target : qui est responsable de gérer le geste ?
 - Action : Quelle action doit-on effectuer quand le geste est reconnu ?
 - View   : Quelle vue doit détecter le geste ?
```

Je créer deux constante : - let upSwipe
- let leftSwipe

Action : va appeler la méthode swipeAndShare qui elle va appeler une autre méthode qui s'appelant wichOrientation() qui renvoie un boolean et renvoie un boolean et en tenant conte de l'orientation de l'écran. Ensuite, elle va appeler la fonction animateSwipe() qui va réaliser l'animation .
Cette application répond au geste sur l'écran et pour cela il est nécessaire de réaliser des connexions il y a deux types :
- les Outlet
- les Actions .

On va connecter la vue et le code exemple : a l'appui du bouton c'est un Outlet (connexion entre une propriété et une vue, je fait référence au label qui est à l'intérieur) .
Et aussi créer une action et la c'est la connexion entre le code et l'interface associée a un événement

Exemple :

Lorsqu'on sélectionne une vue (donc le bouton) on veut bien qu'il se passe quelque chose donc choisi quelle type d'événement je veux qu'il détecte .

Le code suivant :

```
Let upSwipe = UISwipeGestureRecognizer(target: self, action: #selector(swipeAndShare(swipe:)) )
  stackSwipe.addGestureRecognizer(upSwipe)
  upSwipe
```

Instagrid est compose de deux actions qui sont : `@IBAction func` frameButttonThree qui représente les trois boutons du bas .
`@IBAction func` buttonViewCenter(_ sender: UIButton) qui représente les quatre boutons de ma grille centrale .

- Huit méthodes qui vont agir et vont être appeler a des moment précis pour suivre le déroulement de l'application .

Fonctionnalités :

1.Sélection de la disposition des photos :

Ce qui est demande :

En tapant sur l'une de ces dispositions :

La précédente disposition sélectionnée n'est plus marquée comme sélectionnée.

La sélection tapée est marquée comme sélectionnée.

La grille centrale (en bleu foncé) s'adapte en fonction de la nouvelle disposition .

Ce que j'ai réalise :

Pour réaliser cela, j'ai créé ce code :

```
enum layout {

    case one , case two, case three }

    var currentLayout: Layout = .one {

  didSet {

    refreshView()

} }
```

Let image = UIImage(named: Selectedx)

La fonction refreshView() va nous permettre de modifier l'apparence de notre vue en fonction du cadre de vue sélectionné. Pour cela, il faut faire un switch qui définissant les 3 cadre de vue possible qui sont définis dans une énumération en fonction du choix la propriété didSet sera effectué .

Je créé une variable prenant le type de l'énumération est qui est égale par défaut au premier cadre.

Je créé une constante et la classe UIImage à un initialiseur prenant en paramètre le nom de l'image, le nom de l'image qui est une image dans Asset catalog .

Pour cacher certains bouton lors du choix sélectionné j'utilise la propriété : isHidden de UIView de type bool

2. Ajout de photos

Ce qui est demande :

La photo doit être **centrée** , **sans être altérée** (les proportions sont maintenues) et **prendre tout l'espace possible** (pas de "blanc").

Si l'utilisateur clique sur une photo dans la grille, il peut choisir dans la photothèque une nouvelle image pour la remplacer.

Pour permettre de faire ce qui est demandé, on va utiliser des fonctions déjà existantes dans la bibliothèque de Swift. J'utilise UIImagePickerController0, c'est un contrôleur de sélection d'images, son rôle et de gérer les interactions utilisateur et nous fournit les résultats de ces interactions à un objet délégué.

Ce que j'ai réalise :

Je créer une constante qui va gérer les interactions et qui une instance .

```
let imagePickerController = UIImagePickerController()

imagePickerController.delegate = self
```

Pour résumé, je créé une pop up avec UIAlertController0 afin d'afficher des boîtes de dialogue d'alerte pour l'utilisateur.

Je créé une constante appelée: actionSheet et lui affecte une instance de UIAlertController et qui prend 3 arguments

```
let actionSheet = UIAlertController(title: , message: , preferredStyle: )
```

Je créé une variable qui est de type UIButon qui, à chaque fois que je vais cliquer sur le bouton, le signe plus une pop up s'ouvre et je choisis l'une des photos.

```
var middleButton : UIButton?
```

Et à l'intérieur de la fonction :

```
middleButton?.setImage(image, for: .normal);

middleButton?.imageView?.contentMode = .scaleAspectFill .
```

3. Swipe to share :

Ce qui est demande :

L’utilisateur peut partager la création qu’il vient de réaliser. Pour cela il peut réaliser un swipe vers le haut (en mode portrait) ou vers la gauche (en mode paysage). Le swipe lance une animation faisant glisser la grille principale vers le haut (ou vers la gauche) jusqu'à disparaître de l'écran. Une fois l'animation terminée, la vue UIActivityController s'affiche et permet à l'utilisateur de choisir son application préférée pour partager sa création.

Ce que j’ai réalisé:

Pour créer une animation, on utilise la méthode de classe animate de UIView et prend des paramètres donc voici le code de la fonction
Func animateSwipe(){

```
 UIView.animate(withDuration: 0.6, animations: {self.viewToShare.transform = CGAffineTransform(translationX: 0, y: -850)})
}
```

withDuration : Durée de l’animation
Animations: je suis dans une fermeture, ainsi j’utilise self et viewToShare et c’est sur cela que je veux créer l’animation donc c’est ma grille centrale
CGAffineTransform : - la position horizontale qui est l’axe des X
- La position verticale qui est l’axe des Y
Lorsque je déplace vers le bas les Y vont augmenter
Lorsque je déplace vers la droite la valeur de X va augmenter .

Lorsque l’animation terminé, il faut donner un moyen a l’utilisateur de déclenche UIActivityViewController.
Elle se déclenche au toucher du swipe ou de la flèche que j’ai rassemblée dans une stakView lorsque l’utilisateur fait glisser la vue centrale, l’animation s’effectue et au retour de la grille la pop up s’ouvre .

Avec quelques lignes de code, ma fonction swipeAndShare() peut envoyer des photos via AirDrop, sauvegarder l’image ou la partage via les réseaux sociaux (voir par email) .
On réalise cette action en deux étapes :

- La première crée une constante avec :
let shareActivity = UIActivityViewController()
UIActivityViewController qui est la méthode de partage de contenu d’iOS avec d’autres applications et services .

- La 2ème des étapes :
la méthode imageConversion() qui permet de rendre n’importe quelle UIView dans un UIImage .

Cette application fonctionne parfaitement et respecte le cahier des charges elle pourra faire l’objet d’un test lors de notre soutenance .