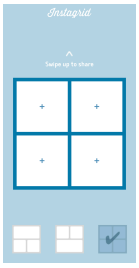
Soutenance P4 -Instagrid

- Description Projet :

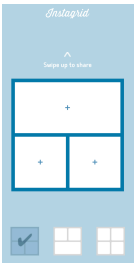
- Nous devions réaliser une application qui permet de combiner de 3 à 4 photos dans un carré qui est ensuite partageable.
- Il y a 3 boutons qui ont chacun une disposition différente, lorsque l'on clique sur un, sa disposition s'affiche sur le carré central.
- Une fois la disposition choisie, on peut cliquer sur les croix à l'intérieur des carrés qui nous dirige vers la librairie de photos de l'iphone en question.
- Une fois la photo choisie, celle-ci se place et se dimensionne automatiquement dans le carré ou l'on a cliqué préalablement.
- Une fois que l'on a choisi les photos que l'on voulait, on peut partager notre combiner de photos grâce à un swipe up (quand le téléphone est en mode portrait) ou un swipe left (quand le téléphone est en mode paysage).
- Quand on swipe, le carré principale monte vers le haut, l'ecran de partage s'affiche.
- Une fois que l'on a partagé la fenêtre se ferme et la carré réapparaît.

P4 - Instagrid

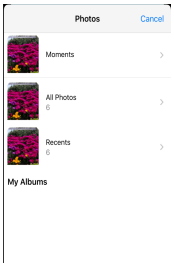
Ouverture de l'application avec cette configuration initialisée dans viewDidLoad Grâce à la func selectLayout



Grâce aux IBActions et à la func selectLayout lorsqu'on clique sur un des 3 boutons, la vue principale change de disposition et le bouton en question est séléctionné



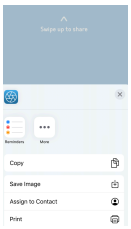
Lorsqu'on clique sur un des plus(+) L'IBAction prend en charge le moyen d'accéder à la librairie de Photo de l'Iphone



P4 - Instagrid

Une fois la photo selectionné,
avec la func imagePickerController
elle s'affiche dans le carré qui a
été cliqué préalablement et sans
Transformer les dismensions.

Lorsqu'on veut partager notre vue
grace au SwipeGestureRecognizer,
la mainView fait une translation et
disparaît, La fenêtre pour partager
s'affiche (UIActivityControler)..
Une fois le partage fait,
La viewMain revient



Ici on gère l'orientation de l'écran,
Si il est en mode Landscapre, le swipe up
devient swipe left, l'animation de la viewMain
Partira vers la gauche

- Explication Code :

D'abord, pour cette application nous utilisons pas le modèle MVC mais seulement view-controller car pour ce projet il n'y a pas de logique c'est majoritairement de l'esthétique.
Ici il y a seulement un dialogue view-Controller.

Pour commencer il a fallu créer des outlets pour toutes les stackviews, boutons, label, view pour pouvoir gérerleurs propriétés dans le codes comme changer le texte du label, ajouter une image au bouton…
Ensuite on a du créer les actions pour gérer la gestion du clique sur les boutons pour changer l'interface de la vue principale et l'ajout d'image.

- **@IBAction func didTapLayoutButton(_sender : UIButton)**

Elle permet de gérer l'interface de la vue principale en cliquant sur un des 3 boutons.
A l'interieur on appel la func selectLayout qui contient un switch avec 3 cas.
En fonction des cas, on doit afficher la vue principale en rapport avec le bouton sélectionné, afficher que le bouton est sélectionné et désélectionné les autres boutons.

- **@IBAction func didTapPhotoButton(_ sender: UIButton!)**

Cette fonction prend en charge un delegate (un delegate : permet de nous servir de protocole, elle *délègue* la tâche d'afficher le résultat à d'autres objets - à condition que ces objets soient conformes au protocole exigé par le générateur, ici déjà créer par les équipes d'apple).

Cette fonction vérifie si la librairie de photo est disponible à l'ouverture et si on peut y accéder. Enfin nous présentons le contrôleur de photothèque avec une animation standard du bas de l'écran.

Nous avons du créer plusieurs propriétés :

- screnHeight : qui prend hauteur de l'écran. Utile pour l'animation
- currentStyle : nombre entier. Utile pour la sélection des layouts
- isLandscape : un booléen qui nous servira pour savoir si l'écran est en mode paysage ou portrait qui prend aussi en charge les iOS supérieur à 13 ou inférieur car notre application doit fonctionner sur les iOS 11.

- **@Objc Func manageOrientation()**

Dans cette fonction on gère l'affichage du label en mode portrait ou paysage, si le mode paysage est détecté grâce à la variable booléenne **isLandscape**, on affichera « swipe left to share » sinon « swipe up to share »
Et on utilise la fonction **selectLayout()** pour ne pas changer le bouton sélectionné en mode paysage ou portrait.

- **@Objc func animeViewMain()**

Ici nous faisons une animation de la v**iewMain** d'une durée de 0,5scd d'une translation sur l'axe des - X en mode paysage de la taille de l'écran grâce à la variable **screenHeight** (pour être sur que la vue disparaisse) et sur l'axe Y en mode portrait.

- **Private func shareImage()**

Cette fonction permet de transformer l'**UIView** en image pour pouvoir la partager ensuite. Dans un premier temps on déclare une variable qui prend en élément les dimensions de l'**UIView** et ensuite on transforme cette variable en image avec la déclaration « **drawHierarchy** ».

Ensuite on partage avec le contrôleur **UIActivityViewController** notre image créer précédemment. Le paramètre **activityItems** prend en charge un tableau de données, c'est pourquoi nous avons créer une variable « items » qui a pour valeur un tableau qui contient la variable image créer précédemment.
Le « **completionWithItemHandler** » permet de fermer la fenêtre de partage une fois que l'utilisateur à fini de se servir de l'**UIActivityViewController**.
Par la suite nous avons rajouter le « **self.viewMain.transform = .identity** » pour faire revenir la vue à sa position initiale une fois le partage terminer.

- **Func imagePickerController()**

Dans cette fonction on dit que l'imageView que l'on sélectionne doit garder sa dimension pour ne pas qu'elle soit élargie quand l'espace est plus grand.
Et que l'on doit placer l'image sélectionné « **selectedButton** » dans le bouton ou l'on a cliqué auparavant.

- **@Objc dunc manageSwipe(sender : UISwipeGestureRecognizer)**

Dans cette fonction on veut que le sender ait reçu une direction .left ET que le téléphone soit en mode landscape ou que mon sender ait reçu une direction .up et que le téléphone soit pas en mode portrait.
Les directions ont été configurés dans le viewDidLoad :

```
let up = UISwipeGestureRecognizer(target: self, action: #selector(manageSwipe))
let left = UISwipeGestureRecognizer(target: self, action: #selector(manageSwipe))
up.direction = .up
left.direction = .left
swipeUp.addGestureRecognizer(up)
swipeUp.addGestureRecognizer(left)
```

Ensuite on appel les fonctions animeViewMain et shareImage

- **Func selectLayout(style :Int)**

Grâce au switch, on gère la gestion des layouts, et de la vue principale.
Quand on clique sur un layout, ça cache un bouton du haut ou non, ça affiche le layout sélectionné et désélectionne les autres layouts.