

Projet 5

Instagrid

Parcours DA IOS
Daniel BENDEMAGH

Objectif

Dans le cadre du développement du projet 5 Instagrid, il m'a été demandé d'ajouter un ou plusieurs bonus à l'application.

Bonus

Bonus 1 - Appareil photo

En plus de la sélection d'une image dans la photothèque, on peut choisir de prendre une photo.

MainVC.swift

```
@IBAction func plusButtonTapped(_ sender: UIButton) {  
    let tag = sender.tag  
    imageSelected = tag  
  
    chooseMedia(title: "Add image")  
}
```

La fonction **chooseMedia** affiche une alerte de type `actionSheet`, qui permet de choisir entre la photothèque et l'appareil photo :

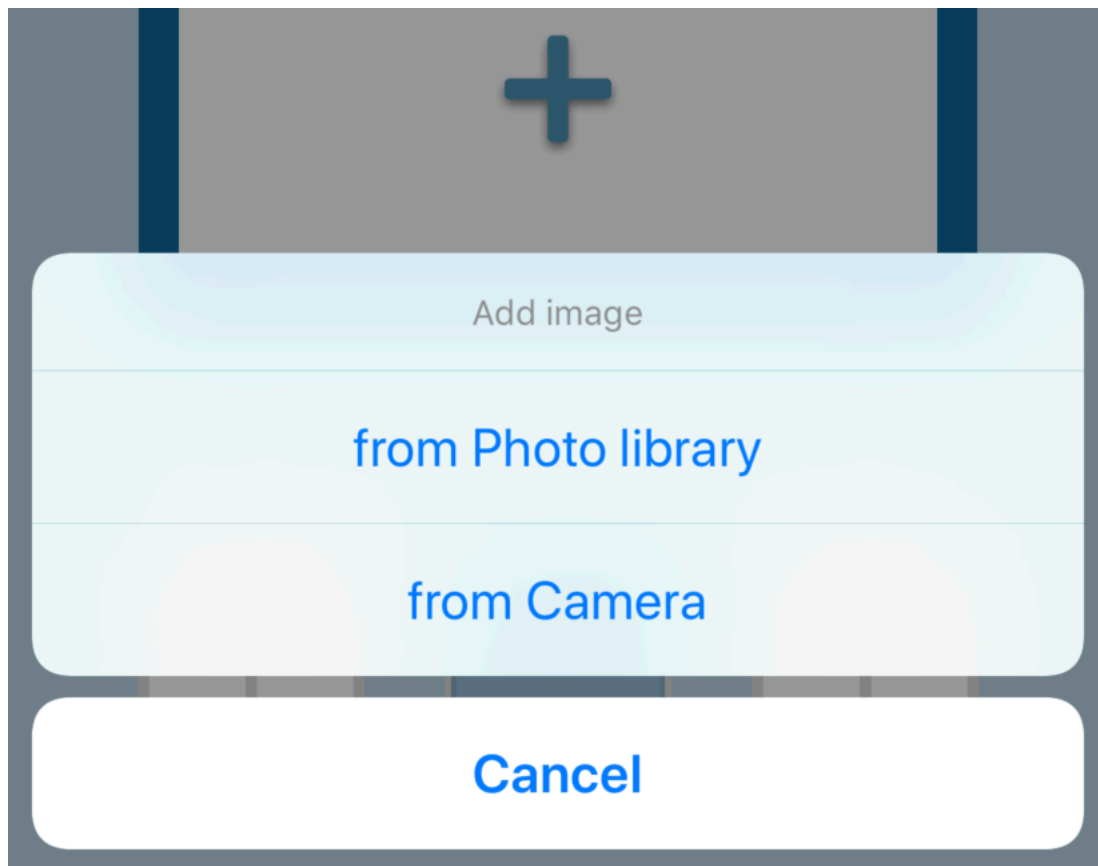
MainVC.swift

```
func chooseMedia(title: String) {  
    let alert = UIAlertController(title: title, message: nil, preferredStyle: .actionSheet)  
  
    alert.addAction(UIAlertAction(title: "from Photo library", style: .default, handler: { (action) in  
        self.getImageFrom(sourceType: .photoLibrary)  
    })))  
  
    alert.addAction(UIAlertAction(title: "from Camera", style: .default, handler: { (action) in  
        self.getImageFrom(sourceType: .camera)  
    })))  
  
    alert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: nil))  
  
    self.present(alert, animated: true, completion: nil)  
}
```

Dans les deux cas, la fonction **getImageFrom** est appelée pour afficher le media correspondant (`photoLibrary` ou `camera`) :

MainVC.swift

```
// Get image from Photo Library or Camera  
func getImageFrom(sourceType: UIImagePickerControllerSourceType) {  
    imagePicker.sourceType = sourceType  
    present(imagePicker, animated: true, completion: nil)  
}
```



Bonus 2 – Changer l'image

Il est possible de changer l'image sélectionnée.

Un UITapGestureRecognizer est ajouté pour chaque UIImageView :

MainVC.swift

```
// Bonus - Select other image
collectionView.photoImageViews.forEach {
    $0.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(imageTapped(gesture:))))
    $0.isUserInteractionEnabled = true
}
```

La fonction **imageTapped** vérifie que le bouton + de l'image est invisible (ce qui signifie qu'une image a été sélectionnée).

Si le bouton est invisible, la fonction **chooseMedia** est appelée.

MainVC.swift

```
@objc func imageTapped(gesture: UIGestureRecognizer) {
    guard let tag = gesture.view?.tag else { return }

    if gridView.plusIsHidden(tag: tag) {
        imageSelected = tag
        chooseMedia(title: "Change image")
    }
}
```

GridView.swift

```
func plusIsHidden(tag: Int) -> Bool {
    return plusButtons[tag].isHidden
}
```

Bonus 3 – Changer la couleur de la grille

La couleur de la grille peut être changée en double tapant sur la couleur en background.

Les couleurs (Color literal) sont stockées dans un tableau et une variable colorIndex est utilisée pour pointer la couleur à afficher :

GridView.swift

```
let backgroundColors = [■, ■, ■, ■, ■]
var colorIndex = 0
```

Un UITapGestureRecognizer est mis en place pour la double tape :

MainVC.swift

```
// Bonus - Modify Gridview backgroundcolor with double tap
let TapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(gridviewDoubleTapped(gesture:)))
TapGestureRecognizer.numberOfTapsRequired = 2
gridView.addGestureRecognizer(TapGestureRecognizer)
```

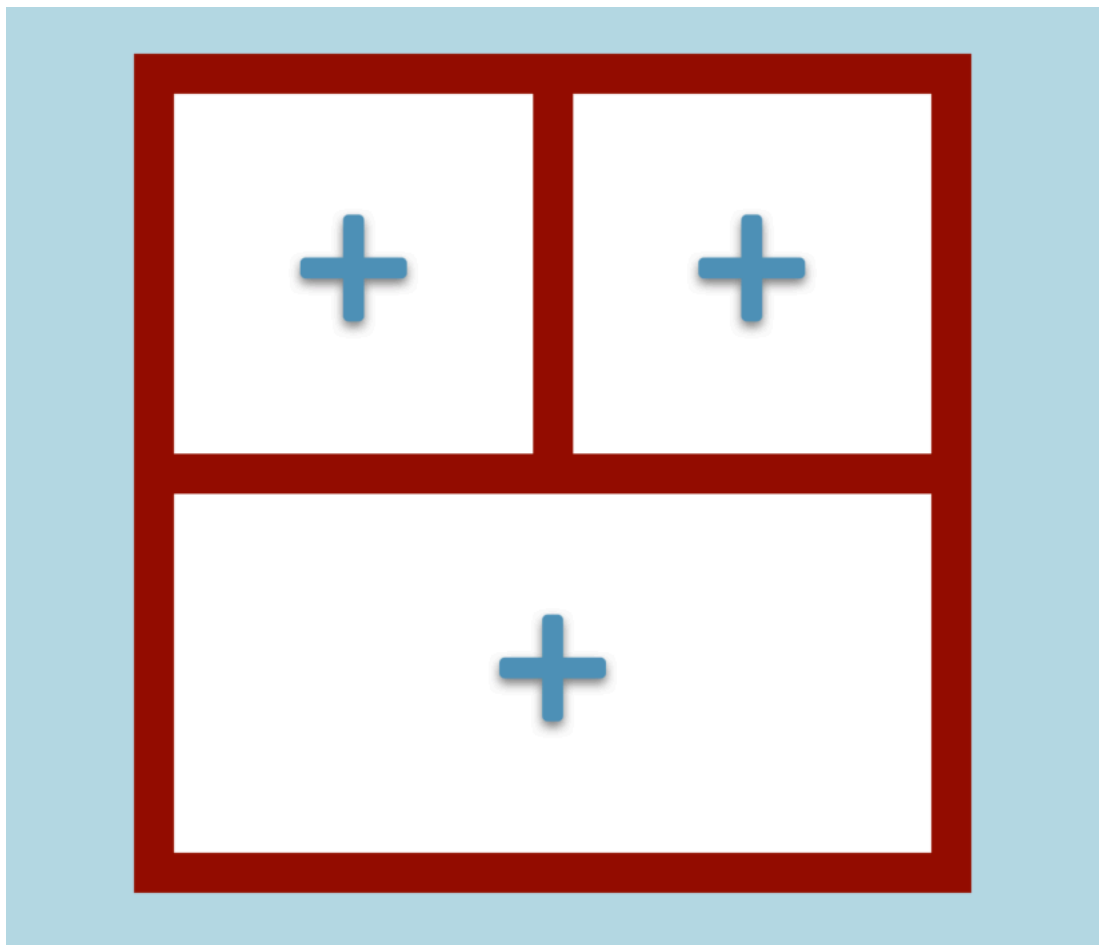
MainVC.swift

```
@objc func gridviewDoubleTapped(gesture: UITapGestureRecognizer) {
    gridView.setBackgroundColor()
}
```

Pour gérer le changement de couleur, la fonction **nextBackgroundColor** renvoie la couleur suivante contenue dans le tableau et repart à zéro dès qu'elle a atteint la fin du tableau.

GridView.swift

```
func setBackgroundColor() {  
    backgroundColor = nextBackgroundColor()  
}  
  
private func nextBackgroundColor() -> UIColor {  
    colorIndex += 1  
    if colorIndex > backgroundColors.count - 1 {  
        colorIndex = 0  
    }  
  
    return backgroundColors[colorIndex]  
}
```



Bonus 3 - Effacer la grille

Il est possible d'effacer la grille en faisant un swipe vers le bas pour le mode portrait et vers la droite pour le mode paysage.

Un UISwipeGestureRecognizer est déclaré :

```
private lazy var eraseSwipeGR: UISwipeGestureRecognizer = UISwipeGestureRecognizer(target: self, action: #selector(eraseEvent))
```

Le Swipe gesture recognizer devra aller dans le sens inverse du swipe de partage. la fonction **deviceOrientationDidChange** gère la direction du swipe (pour le partage et pour l'effacement) :

MainVC.swift

```
@objc private func deviceOrientationDidChange() {
    switch UIDevice.current.orientation {
    case .portrait:
        gridView.setLabelText(text: "Swipe up to share")
        shareSwipeGR.direction = .up
        eraseSwipeGR.direction = .down
    case .landscapeLeft, .landscapeRight:
        gridView.setLabelText(text: "Swipe left to share")
        shareSwipeGR.direction = .left
        eraseSwipeGR.direction = .right
    default:
        break
    }
}
```

Au moment du swipe pour effacer, une animation fait disparaître la grille (comme pour le partage mais dans le sens inverse).

Dans la fonction **eraseGrid**, je vérifie si la grille est vide (fonction **gridIsEmpty**) :

Si elle est vide, seule la couleur du background est réinitialisée.

Si elle n'est pas vide, une alerte sera affichée pour demander la confirmation de la suppression.

GridView.swift


```
func eraseGrid() {
    if isEmpty() {
        // Grid empty, no alert message needed
        resetGrid()
    } else {
        displayAlertDelegate.showEraseAlert(title: "Erase", message: "Do you want to erase grid ?")
    }
}

// Verify if the grid contain at least one image
private func isEmpty() -> Bool {
    for photo in photoImageViews {
        if let superview = photo.superview {
            // If Parent View not hidden and image not nul, grid is not empty
            if !superview.isHidden && photo.image != nil {
                return false
            }
        }
    }

    return true
}

func eraseImages() {
    for photo in photoImageViews {
        photo.image = nil
    }

    for button in plusButtons {
        button.isHidden = false
    }
}

func resetGrid() {
    eraseImages()
    // Original background color
    backgroundColor = 
    colorIndex = 0

    UIView.animate(withDuration: 0.3, animations: {
        self.transform = .identity
    })
}
```

Pour le message d'alerte, j'utilise un Protocol/Delegate entre la vue et le controller :

Gridview.swift

```
protocol GridViewDelegate {
    func showEraseAlert(title: String, message: String)
}

class GridView: UIView {

    var displayAlertDelegate: GridViewDelegate!
```

MainVC.swift

```
extension MainVC: GridViewDelegate {
    // Question alert before erase grid
    func showEraseAlert(title: String, message: String) {
        let alert = UIAlertController(title: title, message: message, preferredStyle: .alert)
        alert.addAction(UIAlertAction(title: "Yes", style: .default, handler: { (action) in
            alert.dismiss(animated: true, completion: nil)
            self.gridView.resetGrid()
        }))
        alert.addAction(UIAlertAction(title: "No", style: .cancel, handler: { (action) in
            alert.dismiss(animated: true, completion: nil)
            self.gridAnimationBack(actionType: .erase)
        }))
        present(alert, animated: true, completion: nil)
    }
}
```