Désécot William
Étudiant OpenClassrooms

Parcours Développeur d'application iOS
Projet 7 – CountOnMe
Bonus

Année 2018

Pour le projet 7 du parcours Développeur d'application iOS, il est demandé de d'améliorer une application existante nommé « CountOnMe », une calculette qui prend en charge les opérations tels que l'addition et la soustraction. Une fois l'application modifiée il faut implémenter un bonus.

Pour ce projet, j'ai choisi de rajouter une fonctionnalité mémoire à la calculatrice. Il est donc possible d'enregistrer le résultat d'une opération dans la mémoire pour ensuite réutiliser ce nombre dans les calculs suivants.
Deux boutons ont donc été rajouté au design de l'application comme le montre l'image ci-dessous :

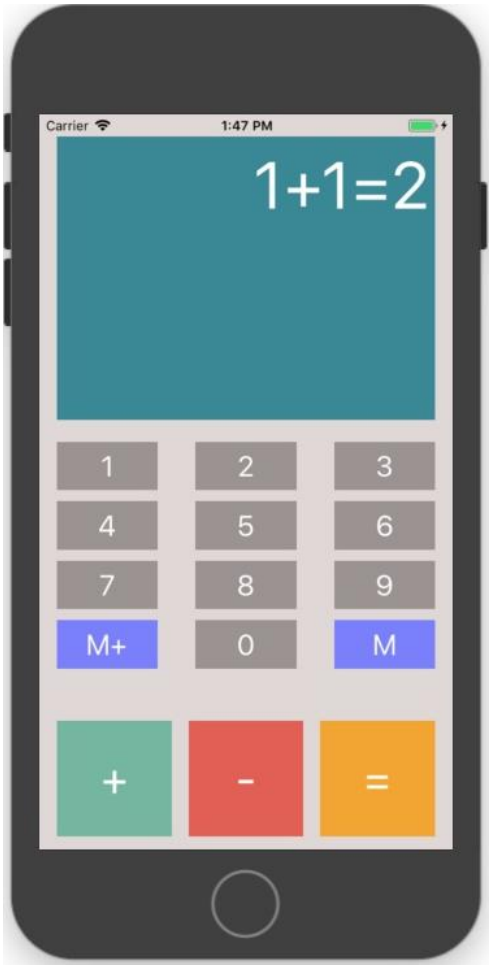

Figure 1 : Design de l'application après avoir rajouté les boutons de mémoire de la calculatrice

Ces deux boutons sont associés à deux fonctions dans le Controller. La fonction tappedAddMemoryButton() qui est associé avec le bouton M+ a pour but d'ajouter le résultat de l'opération précédente à la mémoire de la calculatrice.

```swift
// Bonus
@IBAction func tappedAddMemoryButton() {

    UIView.animate(withDuration: 0.5, delay: 0, usingSpringWithDamping: 0.5, initialSpringVelocity: 0.5, options: .curveEaseIn, animations: {
        self.memoryLabel.transform = self.memoryLabel.transform.translatedBy(x: 0, y: -50)
        self.memoryLabel.alpha = 0
    }) { (_) in
        self.memoryLabel.text = "Memory = \(self.operation.addResultToMemory(self.operation.total))"
        UIView.animate(withDuration: 0.5, delay: 0, usingSpringWithDamping: 0.5, initialSpringVelocity: 0.5, options: .curveEaseOut, animations: {
            self.memoryLabel.transform = .identity
            self.memoryLabel.alpha = 1
        }, completion: nil)
    }
}
```

Figure 2 : Fonction tappedAddMemoryButton() dans le Controller

On fait donc appelle à la fonction addResultToMemory() situé dans le modèle de l'application et qui va prendre en paramètre le total de l'opération précédente.

```swift
// Bonus: Add a memory to the calculator in order to re-use the result of the previous operation
public func addResultToMemory(_ result: Int) -> String {
    memory = result
    return String(memory)
}
```

Figure 3 : Fonction addResultToMemory() dans le modèle

La fonction prend en paramètre un Int et va renvoyer un String. Le paramètre est associé à la variable memory qui représente la mémoire de la calculatrice. C'est cette variable qui est ensuite convertie en String et que va renvoyer la fonction.

Une fois la valeur stockée dans la mémoire de la calculatrice, il est possible de réutiliser cette valeur dans un autre calcul. Il suffit pour cela d'appuyer sur le bouton « M ». Ce bouton est associé à la fonction tappedMemoryButton() dans le Controller qui fait appelle à la méthode addMemoryToOperation() dans le modèle.

```swift
@IBAction func tappedMemoryButton() {
    textView.text = operation.addMemoryToOperation()
}
```

Figure 4 : Fonction tappedmemoryButton() dans le Controller

La fonction addMemoryToOperation() renvoit la fonction addNewNumber(memory) qui permet d'ajouter la valeur de la variable memory au tableau stringNumbers et d'afficher la valeur sur la calculatrice.

```swift
public func addMemoryToOperation() -> String {
    return addNewNumber(memory)
}
```

Figure 5 : Fonction addMemoryToOperation() dans le modèle

Pour afficher la valeur enregistrer en mémoire un label a été créé grâce à du code :

```
var memoryLabel: UILabel = {
    let label = UILabel()
    label.text = ""
    label.textColor = .white
    label.numberOfLines = 0
    label.textAlignment = .left
    label.font = UIFont.systemFont(ofSize: 30)
    label.alpha = 1
    label.adjustsFontSizeToFitWidth = true
    // Enable autolayout
    label.translatesAutoresizingMaskIntoConstraints = false
    return label
}()

//MARK: - Methods
fileprivate func setUpLayout() {
    memoryLabel.bottomAnchor.constraint(equalTo: calculView.bottomAnchor, constant: 0).isActive = true
    memoryLabel.leftAnchor.constraint(equalTo: calculView.leftAnchor, constant: 5).isActive = true
    memoryLabel.rightAnchor.constraint(equalTo: calculView.rightAnchor, constant: 5).isActive = true
    memoryLabel.heightAnchor.constraint(equalToConstant: 50).isActive = true
}
```

Figure 6 : Le label qui permet d'afficher la valeur de la mémoire de la calculatrice

Une fois la valeur enregistrée, elle est affichée sur l'écran de l'application comme le montre l'image ci-dessous :

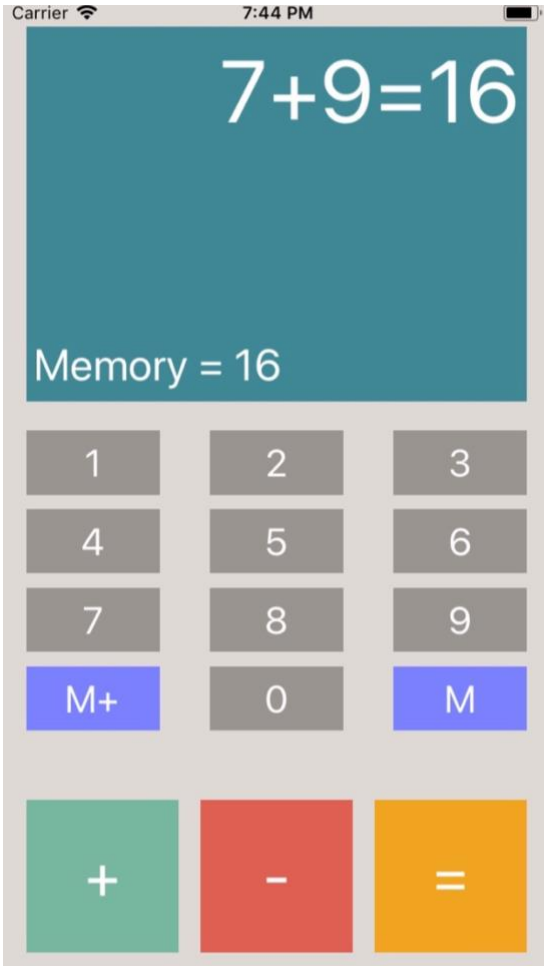

Figure 7 : Écran de l'application après avoir enregistré une valeur en mémoire