Gauthier Akim
Parcours Développeur IOS
Openclassrooms

Projet 7 : CountOnMe

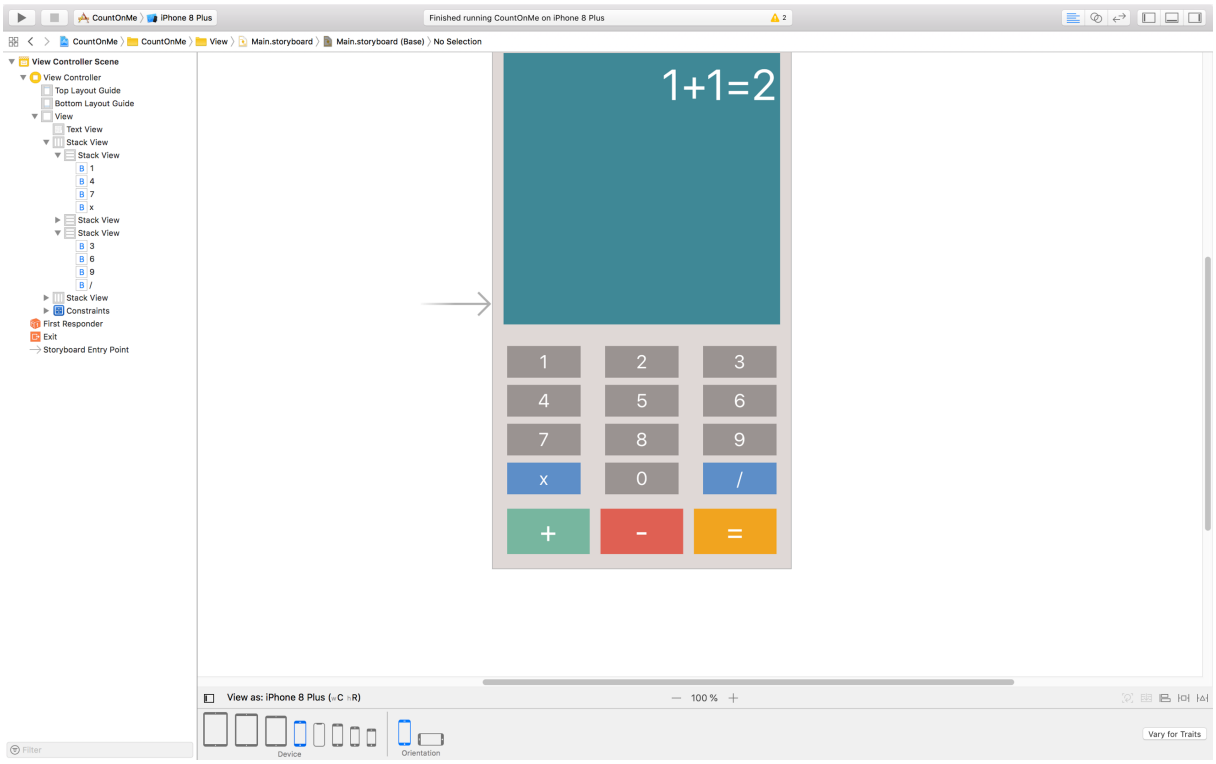
Présentation :

Dans le cadre de la formation Openclassrooms dans le parcours Développeur d'applications IOS. Je présente la feature créer en bonus pour le Projet 7 CountOnMe.

Explication de ma feature :

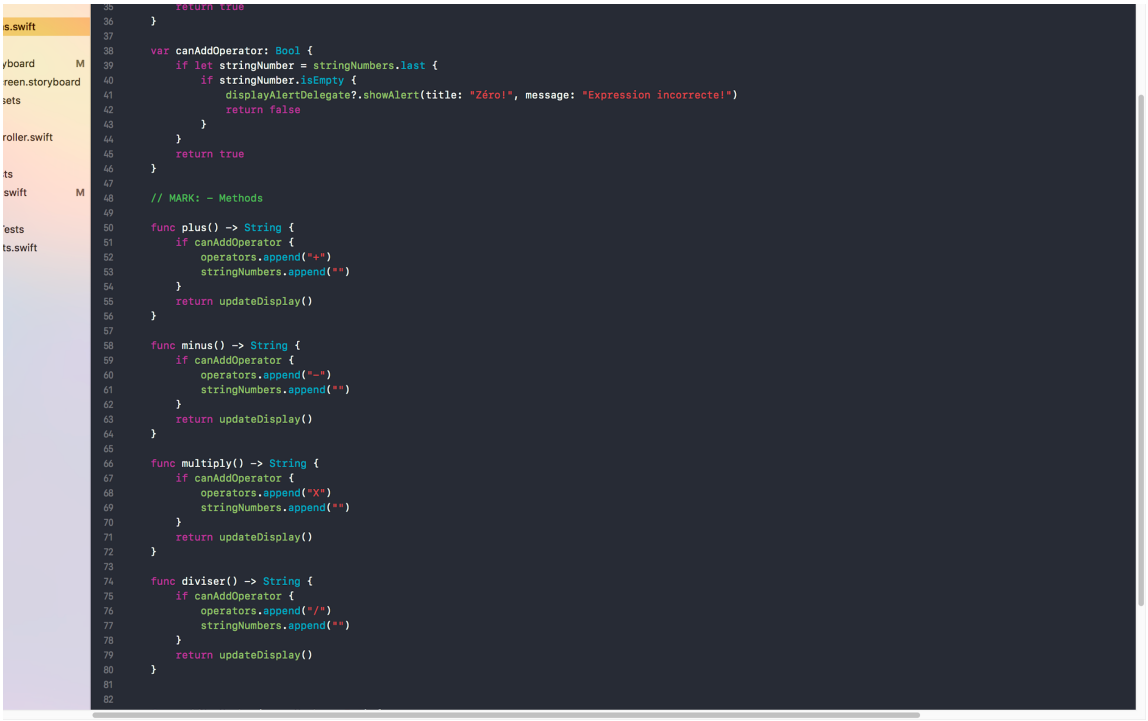
Multiplier et Diviser :

J'ai tout d'abord implémenté les 2 boutons (le Multiplier et le Diviser) à notre Interface dans le Main Storyboard. Puis venir ajouter une action à chacun dans le ViewController.



En revenant dans ma class Operations, j'ai dans ma fonction priority calcul dans calculate total.Qui ajoute plusieurs conditions qui vont vérifier lorsque premièrement si notre tableau de calculs contient bien le multiplier et diviser les calculer en priorité, faire correspondre le calcul de chacun et les additionner ou soustraire si c'est un calcul de plusieurs opérateurs.

Ensuite pour chacun je créer une fonction pour chaque opérateur lorsque celui-ci sera appuyé par l'utilisateur, qui va vérifier que le calcul soit possible, puis l'effectuer et renvoyer la valeur du résultat à notre fonction.



```
    return true
}

var canAddOperator: Bool {
    if let stringNumber = stringNumbers.last {
        if stringNumber.isEmpty {
            displayAlertDelegate?.showAlert(title: "Zéro!", message: "Expression incorrecte!")
            return false
        }
    }
    return true
}

// MARK: - Methods

func plus() -> String {
    if canAddOperator {
        operators.append("+")
        stringNumbers.append("")
    }
    return updateDisplay()
}

func minus() -> String {
    if canAddOperator {
        operators.append("-")
        stringNumbers.append("")
    }
    return updateDisplay()
}

func multiply() -> String {
    if canAddOperator {
        operators.append("X")
        stringNumbers.append("")
    }
    return updateDisplay()
}

func diviser() -> String {
    if canAddOperator {
        operators.append("/")
        stringNumbers.append("")
    }
    return updateDisplay()
}
```

Et pour finir, j'instancie la class Opérations dans le ViewController pour venir appeler chaque fonction dans chacune des actions du contrôleur correspondant à son bouton.

```
// MARK: - Properties
var operations = Operations()

override func viewDidLoad() {
    operations.displayAlertDelegate = self
}

// MARK: - Action

@IBAction func tappedNumberButton(_ sender: UIButton) {
    for (i, numberButton) in numberButtons.enumerated() {
        if sender == numberButton {
            operations.addNewNumber(i)
        }
    }
    textView.text = operations.updateDisplay()
}

@IBAction func plus() {
    textView.text = operations.plus()
}

@IBAction func minus() {
    textView.text = operations.minus()
}

@IBAction func equal() {
    textView.text = operations.calculateTotal()
}

@IBAction func multiply() {
    textView.text = operations.multiply()
}

@IBAction func diviser() {
    textView.text = operations.diviser()
}
}

extension ViewController: DisplayAlert {
    func showAlert(title: String, message: String) {
        let alertVC = UIAlertController(title: title, message: message, preferredStyle: .alert)
        alertVC.addAction(UIAlertAction(title: "OK", style: .cancel, handler: nil))
        self.present(alertVC, animated: true, completion: nil)
```