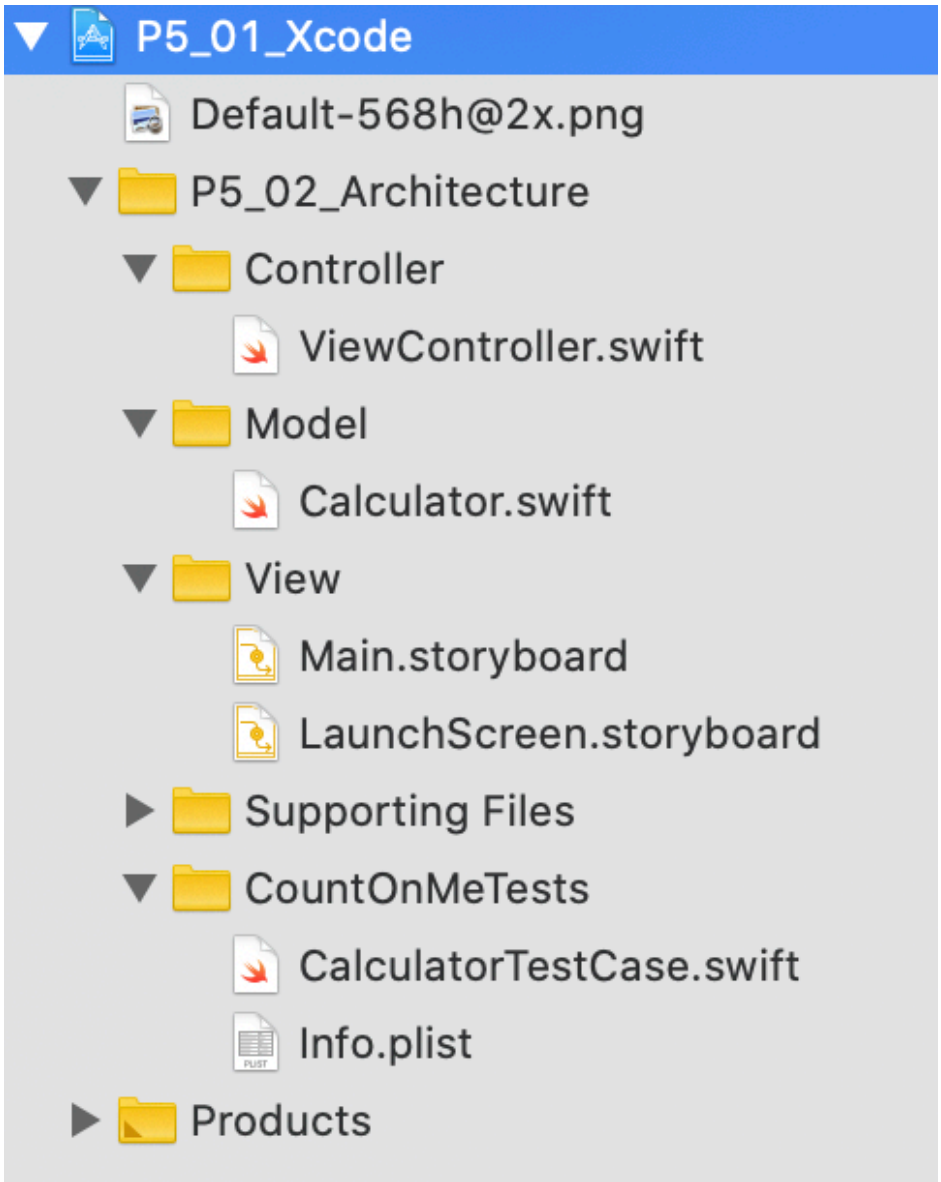
ARCHITECTURE CountOnMe

CountOnMe est une calculatrice iPhone permettant aux utilisateurs d'effectuer les calculs suivant :

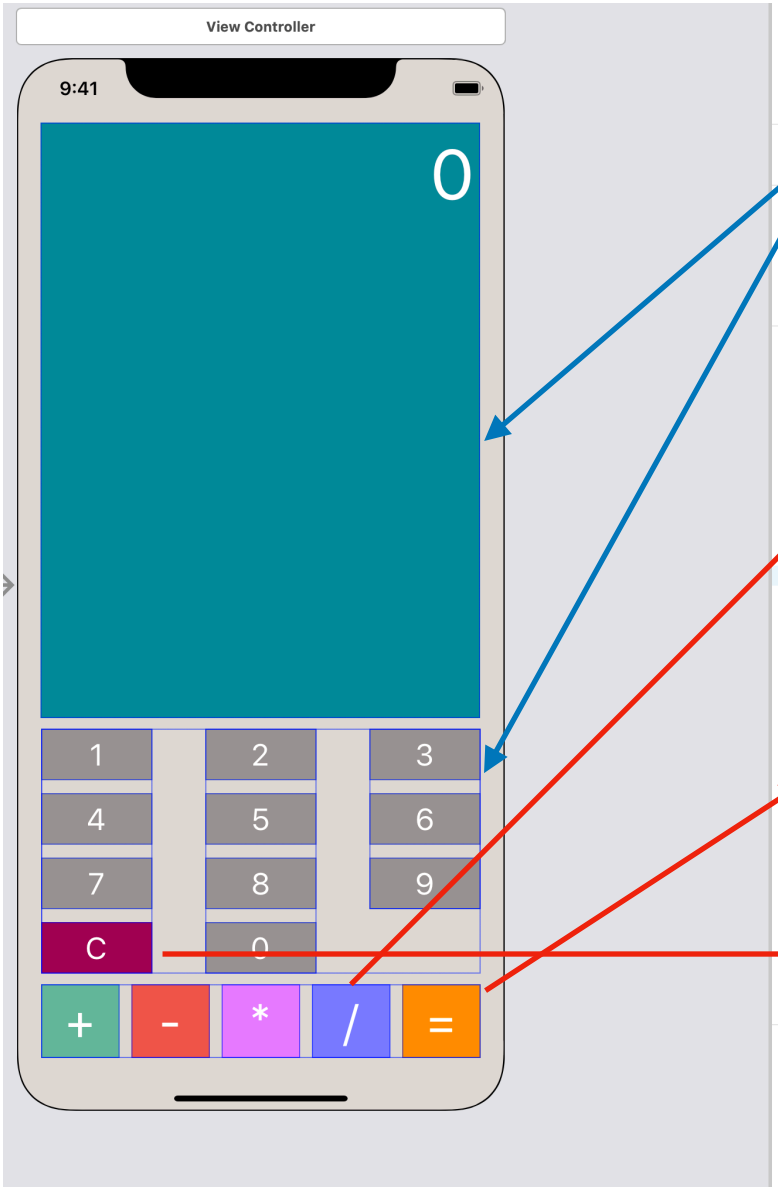
- Addition, soustraction, multiplication et division

L'application inclus la notion de calculs prioritaires (comme la multiplication et la division) lors de calculs à multiples opérateurs.



Notre structure adopte pleinement le modèle MVC (modèle, vue, controller)
L'application comporte majoritairement de la logique (traitement de données, calculs).

Toute la logique (les calculs) sera traité dans la partie du modèle.
Le controller établira la connexion entre le modèle et la vue.
Dans un premier temps, le controller permettra à la vue de transmettre la saisi de l'utilisateur au modèle.
Le controller communiquera ensuite les résultats ou les messages d'erreurs provenant des calculs effectués par le modèle à la vue.
Pour terminer, la vue se chargera d'afficher à l'utilisateur toutes les résultats provenant du modèle par l'intermédiaire du controller.




```swift
//  Copyright © 2019 Vincent Saluzzo. All rights reserved.
//

import UIKit

class ViewController: UIViewController {
    // MARK: - Outlets
    @IBOutlet weak var textView: UITextView!
    @IBOutlet var numberButtons: [UIButton]!
    // MARK: - Properties
    lazy var calculator = Calculator(textViewModel: textView.text)
    // View Life cycles
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }
    // MARK: - Actions
    /// Appuyez sur les boutons numériques
    @IBAction func tappedNumberButton(_ sender: UIButton) {
            guard let numberText = sender.title(for: .normal) else {
                return
            }
            calculator.addNumberForCalculate(number: numberText)
            textView.text = calculator.textViewModel
        }
    /// Appuyez sur les boutons des opérateurs
    @IBAction func tappedOperatorButton(_ sender: UIButton) {
        guard let operatorText = sender.title(for: .normal) else {
            return
        }

        guard calculator.addOpertorForCalculate(operators: operatorText) else {
            return showAlert(with: "Vous avez déjà ajouté un opérateur.")
        }

        textView.text = calculator.textViewModel
    }
    /// Appuyez sur le bouton égal
    @IBAction func tappedEqualButton(_ sender: UIButton) {
        let resultStatus = calculator.makeCalculation()
        guard resultStatus.validity else {
            return showAlert(with: resultStatus.message)
        }
        textView.text = calculator.textViewModel

        }
    /// Appuyez sur le bouton annuler
    @IBAction func tappedResetButton(_ sender: UIButton) {
        calculator.reset()
        textView.text = calculator.textViewModel
    }
    // MARK: - Methods
    /// Message d'alerte
        func showAlert(with message: String) {
        let alertVC = UIAlertController(title: "Erreur :",
                                        message: message,
                                        preferredStyle: .alert)
        alertVC.addAction(UIAlertAction(title: "OK", style: .cancel, handler: nil))
        self.present(alertVC, animated: true, completion: nil)
```

Le fichier CountOnMeTests affichera les tests effectués sur l'ensemble des fonctions de notre modèle et permettra ainsi de mettre en avant le bon fonctionnement de l'application.