



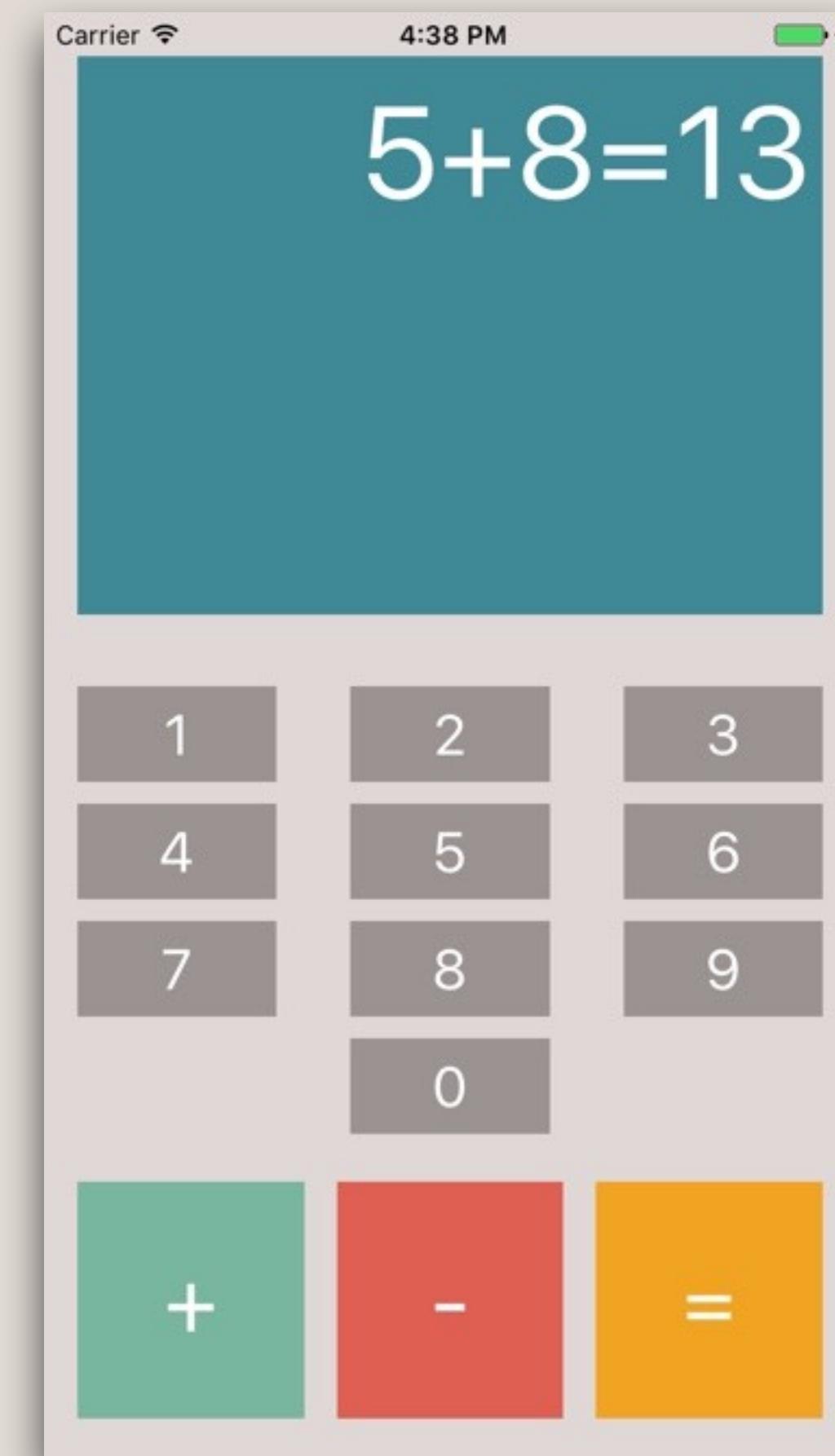
Développer L'APP *CountOnMe*

Count On Me

CAHIER DES CHARGES

Résumé du cahier des charges

- Rendre l'application *responsive*
- Respecter le modèle MVC
- Compléter avec la multiplication et la division
- L'algorithme fourni doit être préservé
- L'usage de l'expression NSExpression n'est pas autorisé
- Tester l'application



Calculatrice CountOnMe, projet fourni

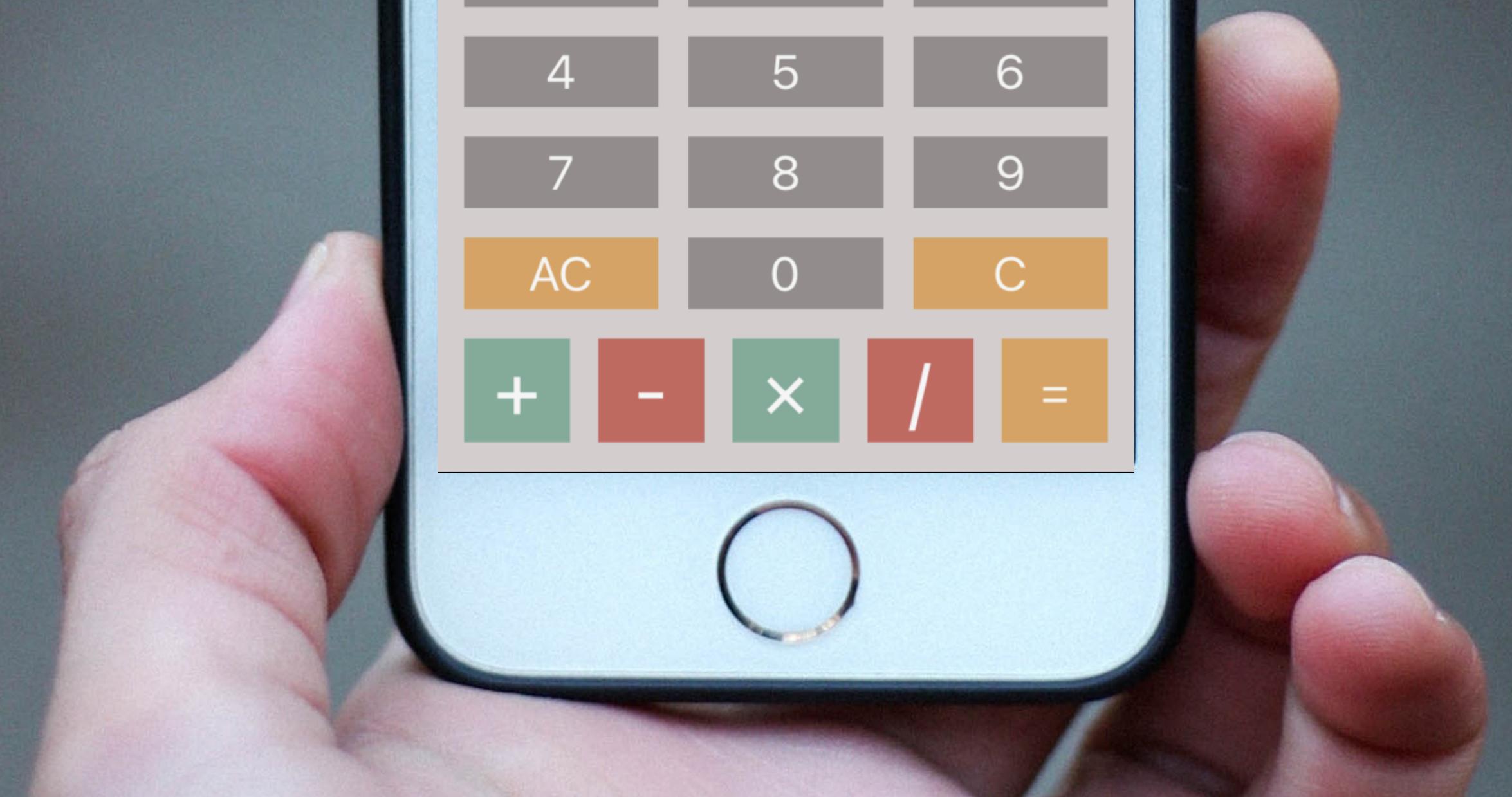
Contraintes techniques

- Le langage utilisé doit être Swift 4 ou supérieur
- Le code doit être commenté correctement et en anglais
- L'application doit être disponible à partir d'iOS 11.0
- L'application est supportée par toutes les tailles d'iPhone (de l'iPhone SE à l'iPhone 13 Pro Max)
- L'application n'a pas à être disponible sur iPad
- L'application supporte l'orientation Portrait

Choisir son opérateur

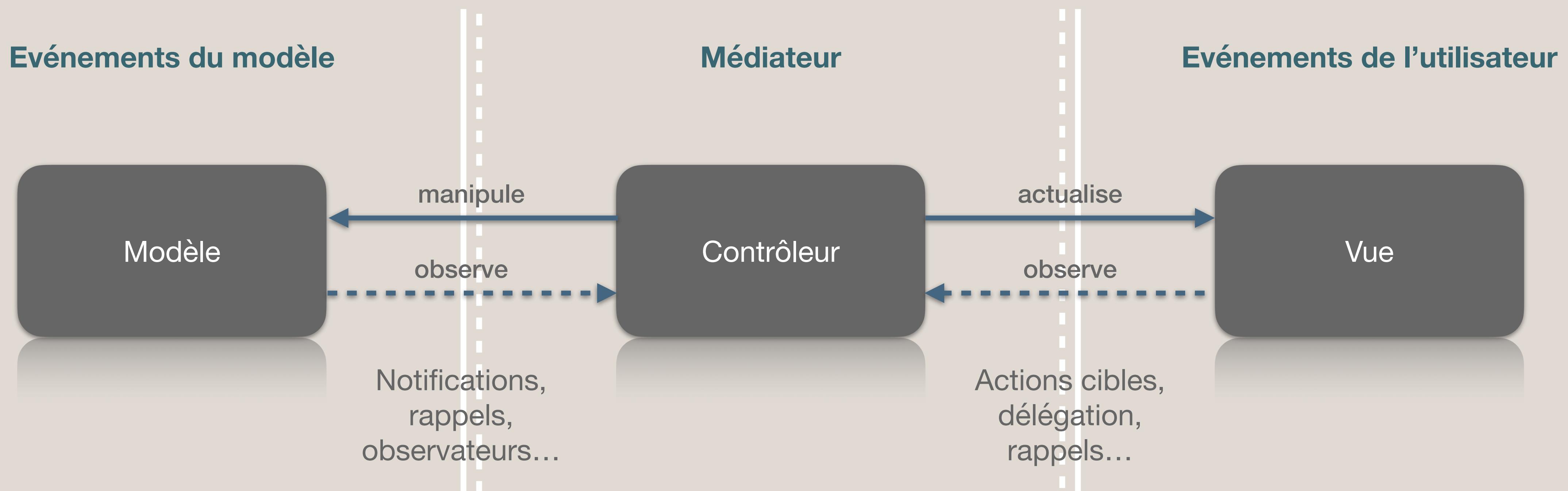
- Choix des opérateurs mathématiques selon la norme International Standard ISO 80000-2



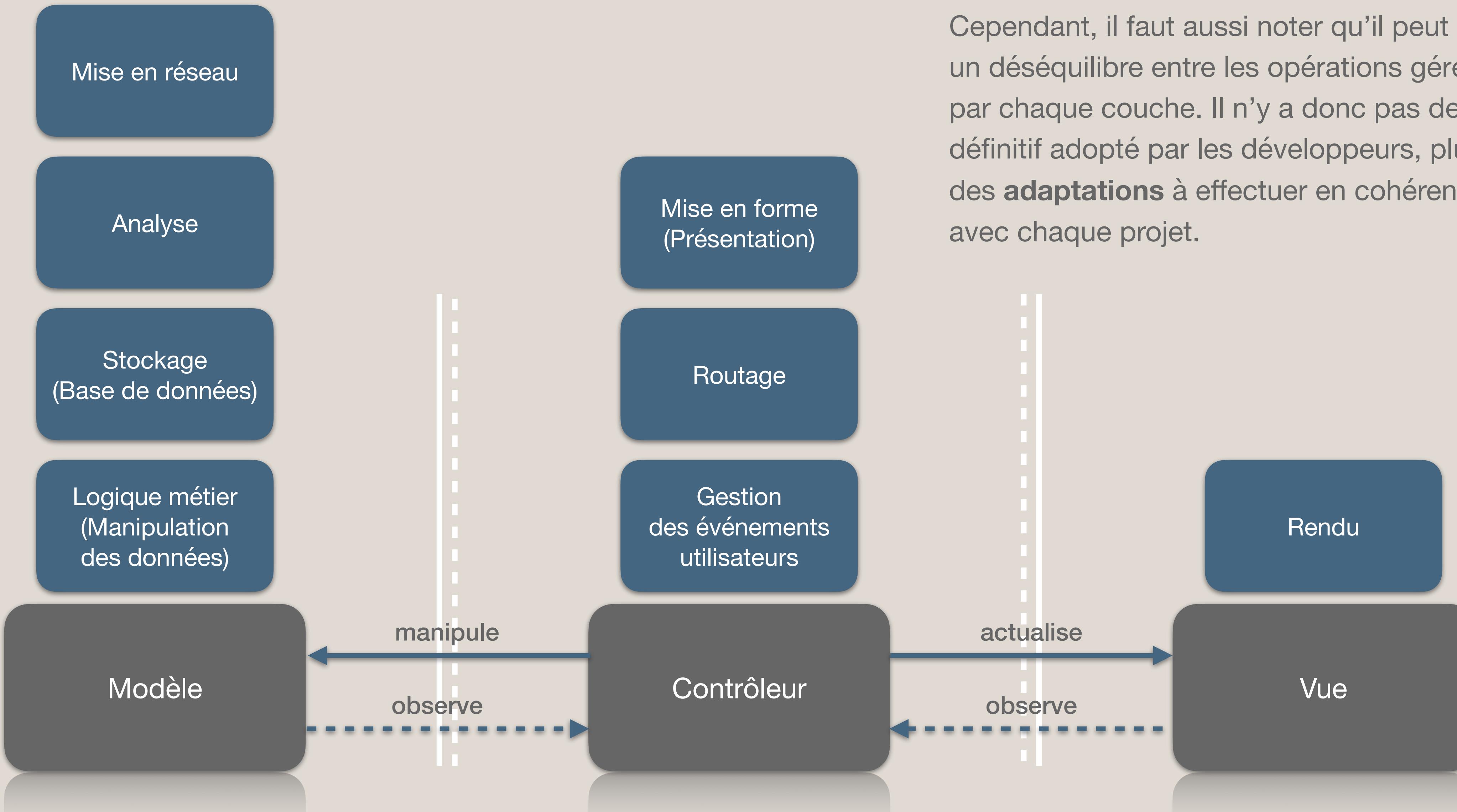


Le modèle
MVC

MVC est un modèle de conception imaginé en 1979 pour décrire le fonctionnement des **interfaces utilisateur**. Mais les écrans d'aujourd'hui sont devenus tactiles, donc les événements de l'utilisateur viennent de la **Vue** elle-même. Ceci a induit un changement majeur dans le MVC d'Apple qui a **redéfini** le modèle pour l'accommorder à ses propres *frameworks* (UIKit, AppKit, etc.).



Un autre changement majeur dans le MVC d'Apple est qu'il devient **un modèle d'architecture** puisque toutes les responsabilités que l'on trouve dans une application sont désormais gérées par une des trois couches du modèle. Il est donc logique qu'Apple recommande l'utilisation de ce modèle pour des applications **simples**, comme c'est le cas pour mon application **CountOnMe**.

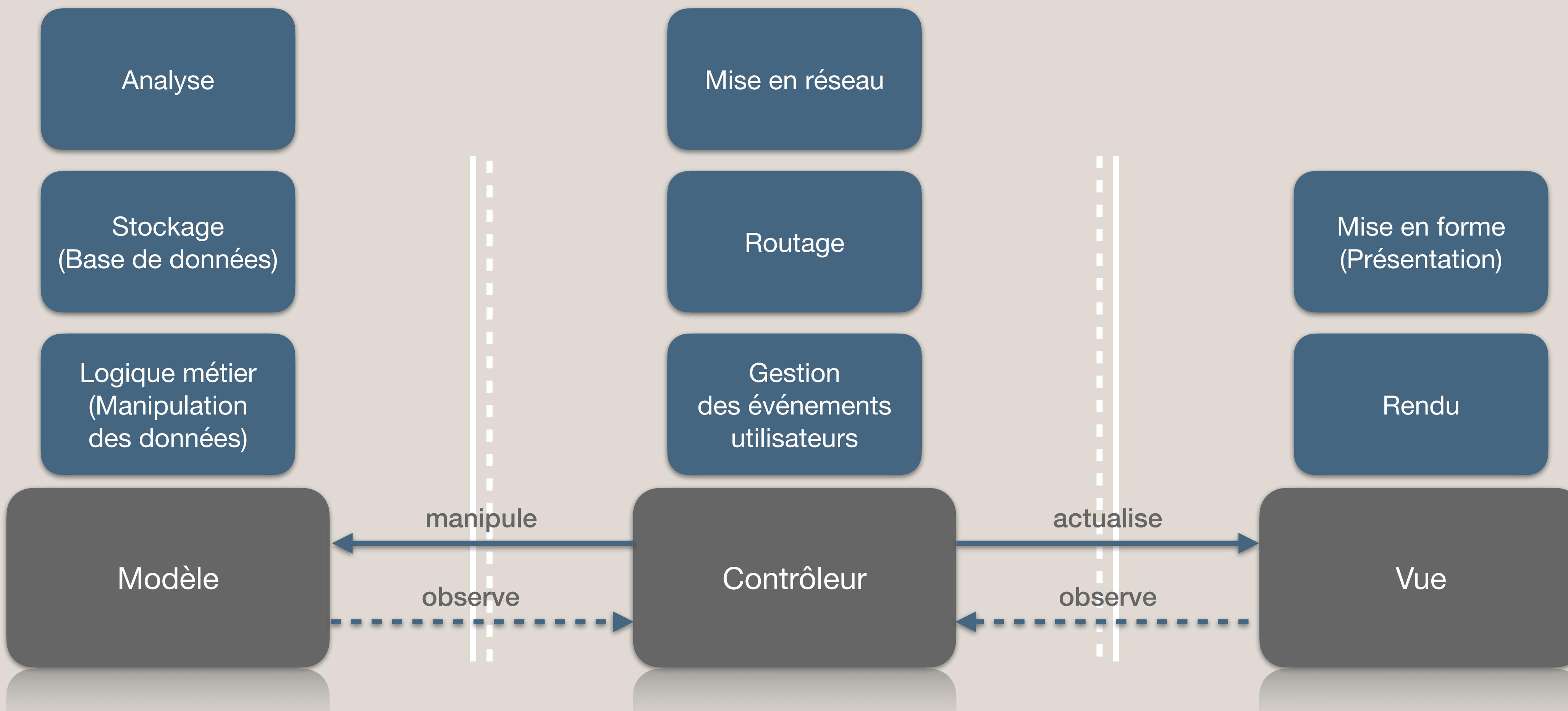


Cependant, il faut aussi noter qu'il peut y avoir un déséquilibre entre les opérations gérées par chaque couche. Il n'y a donc pas de consensus définitif adopté par les développeurs, plutôt des **adaptations** à effectuer en cohérence avec chaque projet.

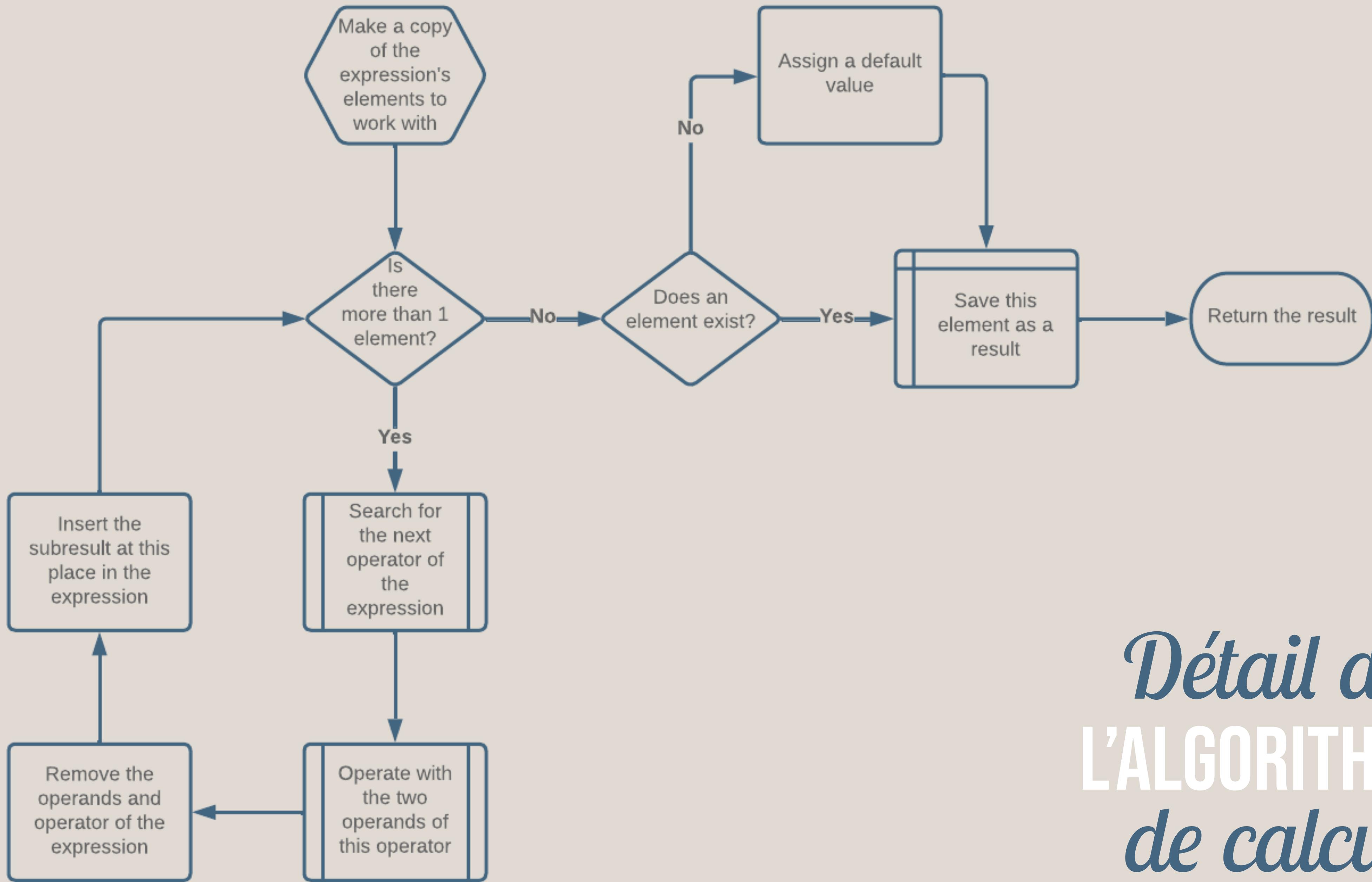
L'une de ces adaptations consisterait aussi à envisager un **rééquilibrage** des responsabilités de chaque couche, comme l'indique Caio Zullo de Essential Developer.

Sur le schéma ci-dessous, les fonctionnalités de mise en forme ont donc été déplacées du Contrôleur vers la Vue et la mise en réseau est passée du Modèle au Contrôleur.

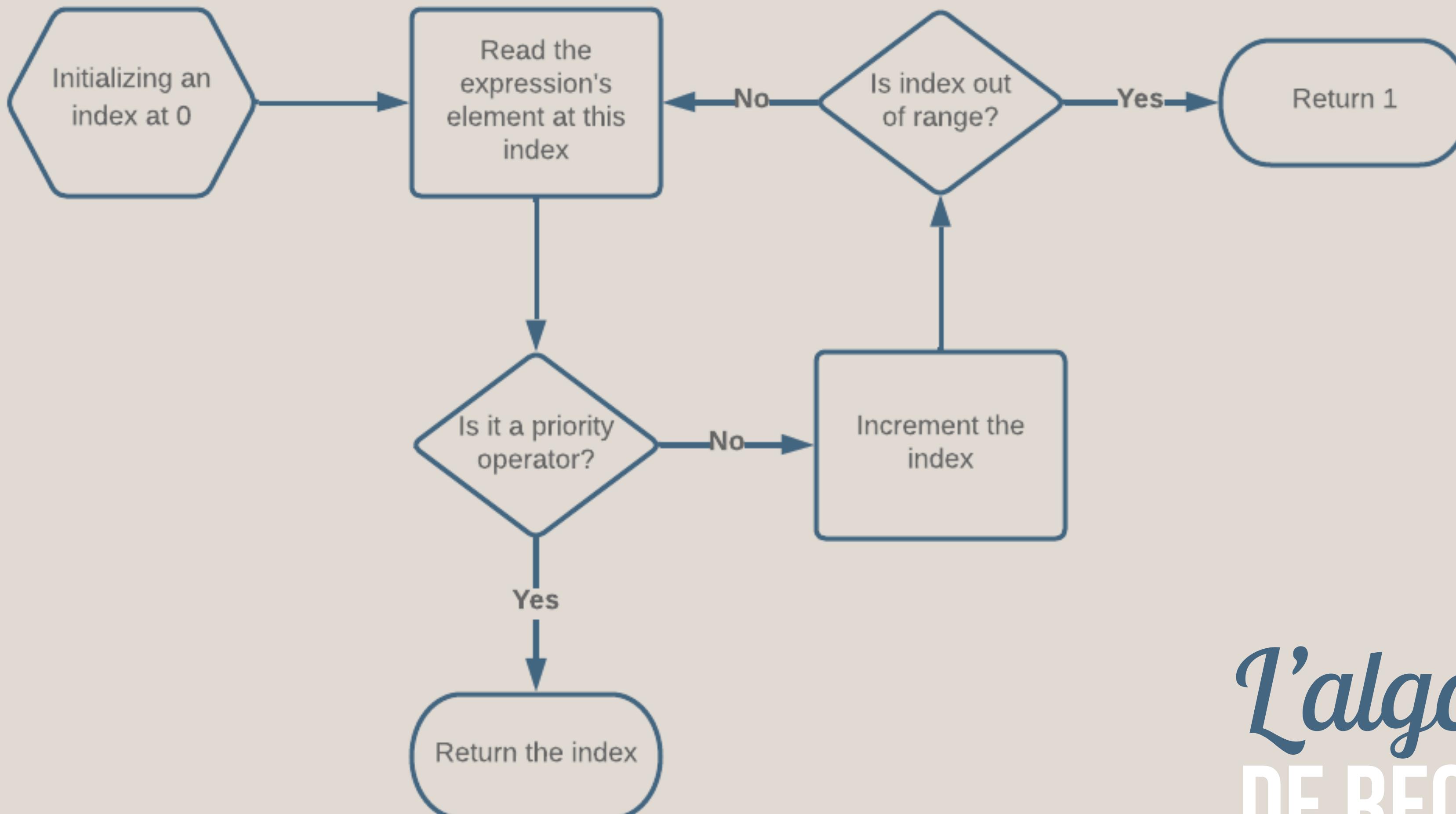
Ces choix discutables conservent une certaine cohérence. Dans tous les cas, c'est la **clarté** du code, son **organisation** et sa **compréhension** par les autres développeurs, qui doivent être privilégiées.



Le fonctionnement par
LES LOGICRAMMES

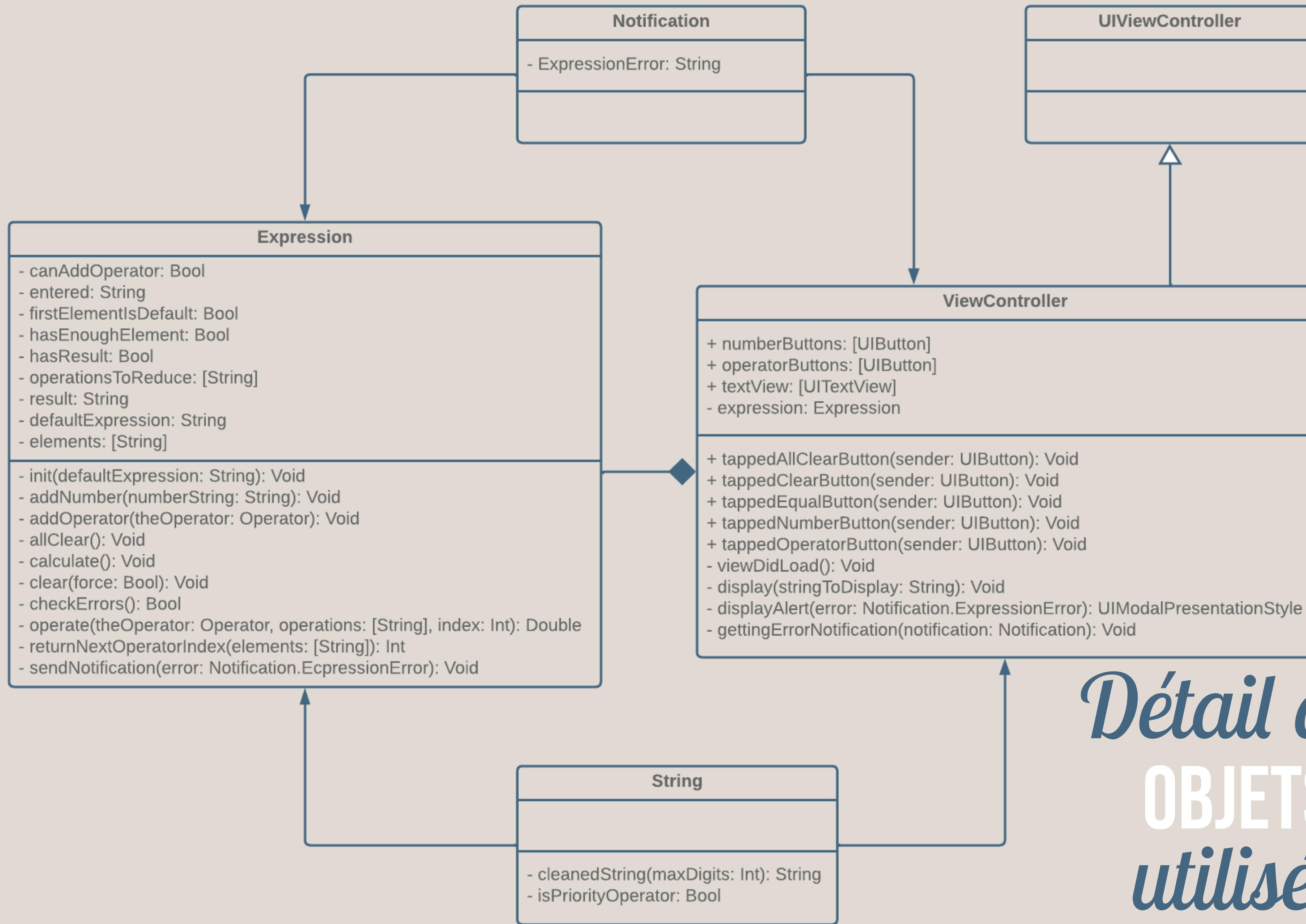


*Détail de
l'ALGORITHME
de calcul*



L'algorithme DE RECHERCHE d'opérateur

Le diagramme
DE CLASSES



*Détail des
OBJETS
utilisés*

Un plongeon
DANS LE CODE

GitHub

- https://github.com/gemini-crocket/Projet_CountOnMe



Le fonctionnement
EN DIRECT

COMPÉTENCES DÉVELOPPÉES SUR CE PROJET

- Algorithmique
- Implémenter l'architecture MVC
- Test Driven Development (TDD)
- Behavior Driven Development (BDD)
- Utiliser le *code coverage* de Xcode
- Utiliser les *snippets* de Xcode

1

CountOnMe
CAHIER DES CHARGES

2

*Le modèle
MVC*

3

*Le fonctionnement par
LES LOGIGRAMMES*

4

*Le diagramme
DE CLASSES*

5

*Un plongeon
DANS LE CODE*

6

*Le fonctionnement
EN DIRECT*

INSPIRÉ PAR

Francis Bloch

Ambroise Collon

Steve Guédon

Paul Hudson

Caio Zullo

Développer L'APP *CountOnMe*

A close-up photograph of a person's hand holding a smartphone. The phone screen displays a slide with large, bold text in French: "Développer L'APP CountOnMe". The background of the slide has a subtle blue gradient with some abstract shapes. The overall composition is centered, with the phone held horizontally.