DEVELOPPEMENT APPLICATION CALCULATRICE

I. INTRODUCTION

Pour débuter le projet, nous devions développer une application de calculatrice sur iOS et Android dans le but de s'initier au développement d'application.

Possédant un appareil sur MacOS et étant plus a l'aise avec les appareils iOS, j'ai préféré commencer avec un développement iOS.

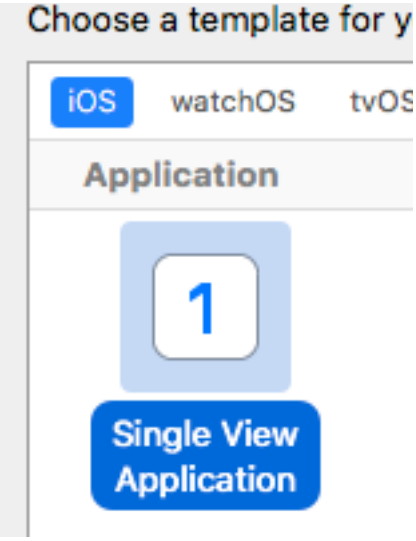
II. XCODE/SWIFT

Xcode est le logiciel de développement de MacOS, il permet de développer pour iOS sur iPad iPhone mais aussi pour l'Apple TV, l'Apple Watch, etc..

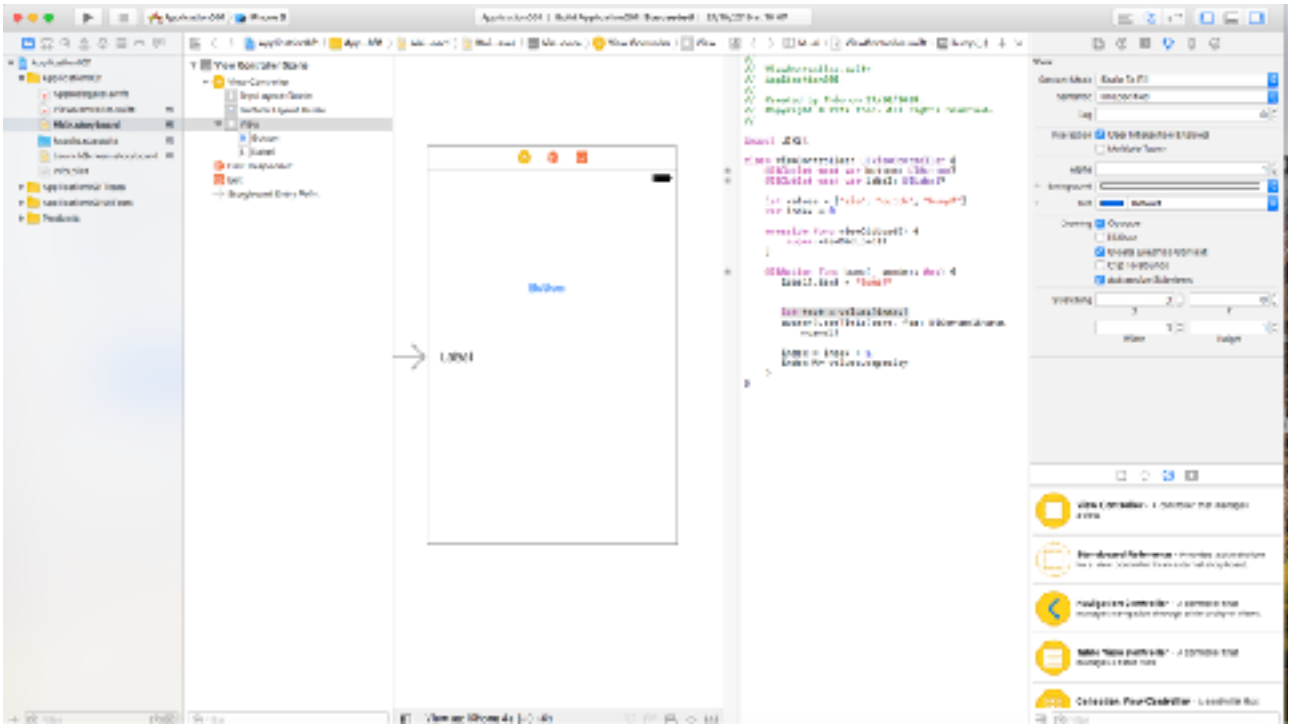
Pour développer sur iOS, j'ai eu le choix entre Objective C et Swift comme langage de programmation, j'ai commencé avec Swift, car il était plus récent et recommandé

1. Découverte de Xcode et de Swift

Pour developper une calculatrice, une application simple a une seule vue suffis, donc j'ai sélectionné un projet en « Single View » .



Voici l'interface de Xcode :



On peut voir que sur l'interface, j'ai intégré deux fenêtre, une pour le contrôleur (a gauche) une pour le code (a droite).

Avec le menu déroulant sur le bas de la droite j'ai placé un bouton et un label, que qui sont configurable sur le menu du haut de la droite.

Puis avec le bouton ctrl et le clique gauche, avec la souris j'ai créé une liaison du bouton vers le code.

```
    @IBOutlet weak var button: UIButton?
    @IBOutlet weak var label: UILabel?
```

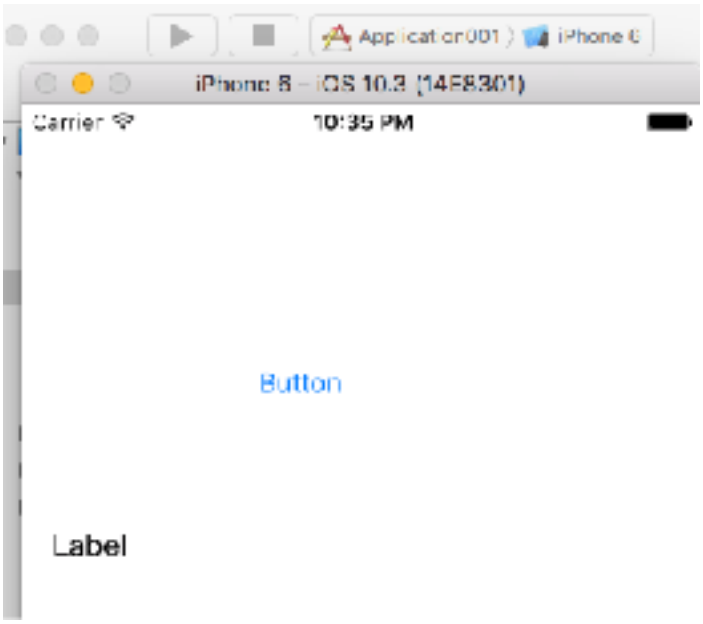
On crée donc des outlets qui vont permettre d'intégrer ces objets dans le code sous forme de variable.

```
@IBAction func bump(_ sender: Any) {
```

On peut utiliser la même manipulation pour intégrer l'action d'un objet sous forme de fonction, pour enclencher un code.

```
@IBAction func buttonClick(_ sender: UIButton) {
    if sender == button0 {
```

On peut aussi lier plusieurs actions de plusieurs bouton a la même fonction, sélectionnera notre bouton activé avec une condition pour le bouton défini.

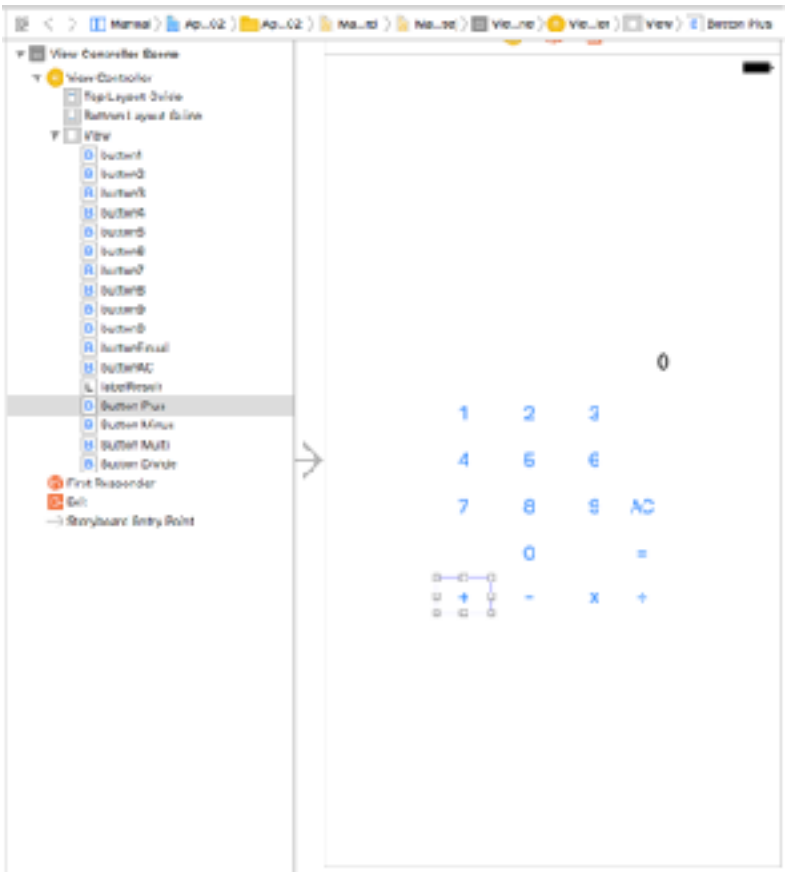


Je peux aussi simuler un iPhone avec Xcode pour tester mon application.

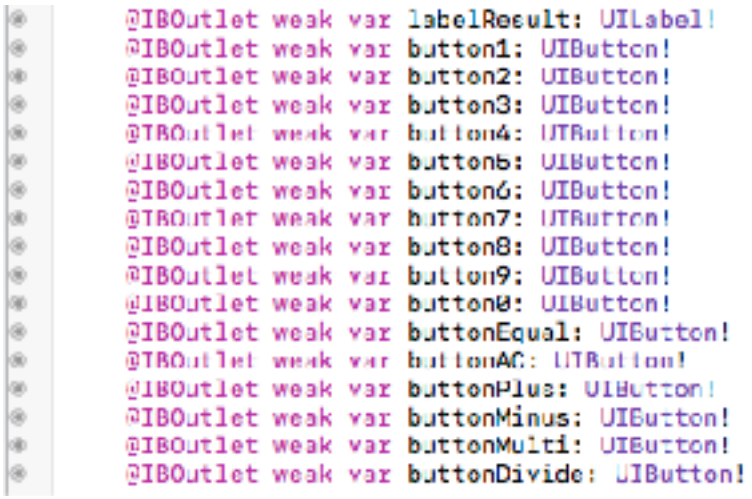
---

2. Développement de la Calculatrice

Ayant acquis quelques notion de base en Xcode et en Swift, je commence le développement de la calculatrice

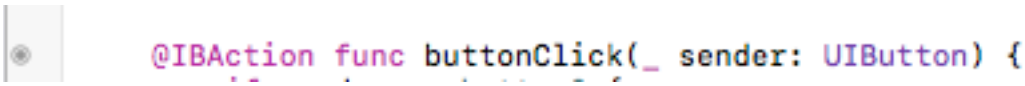


Je commence d'abords par placer les boutons de la calculatrice.



```
@IBOutlet weak var labelResult: UILabel!
@IBOutlet weak var button1: UIButton!
@IBOutlet weak var button2: UIButton!
@IBOutlet weak var button3: UIButton!
@IBOutlet weak var button4: UIButton!
@IBOutlet weak var button5: UIButton!
@IBOutlet weak var button6: UIButton!
@IBOutlet weak var button7: UIButton!
@IBOutlet weak var button8: UIButton!
@IBOutlet weak var button9: UIButton!
@IBOutlet weak var button0: UIButton!
@IBOutlet weak var buttonEqual: UIButton!
@IBOutlet weak var buttonAC: UIButton!
@IBOutlet weak var buttonPlus: UIButton!
@IBOutlet weak var buttonMinus: UIButton!
@IBOutlet weak var buttonMulti: UIButton!
@IBOutlet weak var buttonDivide: UIButton!
```

Puis je les lie au code.



```
@IBAction func buttonClick(_ sender: UIButton) {
```

Ensuite je lie tout les actions des boutons a une fonction qui va s'enclencher lors du click

---

Je commence donc le développement d'un système de calculatrice

Je crée des variable qui me serviront :

```
var affichage = "0"
var operation = ""
var firstNum = 0
var reserve = ""
var SecondNum = 0
var resultat = 0
```

Une pour l'affichage (à 0 au début)
Une qui stockera l'opération
Une pour le premier chiffre de l'opération
Une « réserve » nécessaire pour mon système
Une pour le deuxième chiffre
Et une pour le résultat de l'opération

Puis je commence à écrire dans la fonction

```
else if sender == button1 {
    if affichage == "0" {
        affichage = "1";
    }
    else {
        affichage += "1";
    }
    if firstNum != 0 {
        reserve += "1";
    }
}
```

Je met une condition if pour chaque bouton (si j'appui sur tel bouton), puis je lui dis si l'affichage est a 0, il change le nombre pour le chiffre du bouton, sinon il rajoute le chiffre du bouton après le précédant.
La condition suivante sert a déterminer si le premier nombre a été rentré (plus précisément après l'appui sur un bouton d'un signe d'opération), si c'est le cas, il place le nombre dans une « réserve ».

```
else if sender == buttonAC {
    affichage = "0";
    firstNum = 0
    SecondNum = 0
    operation = ""
    reserve = ""
}
```

Pour le bouton AC, on réinitialise tout à son état de base.

```
else if sender == buttonPlus {
    operation = "+"
    firstNum = Int(affichage)!
```

Pour un bouton de signe d'opération, en plus de l'affichage, nous allons stocker le signe dans une variable, et prendre le chiffre d'affichage pour l'insérer dans une autre variable dédiée (nous le convertissons en INT pour pouvoir l'utiliser dans une opération, car la variable par défaut est en STRING).

```
else if sender == buttonEqual {
    affichage += " = ";
    SecondNum = Int(reserve)!
    if operation == "+" {
        resultat = firstNum+SecondNum
    }
    if operation == "-" {
        resultat = firstNum-SecondNum
    }
    if operation == "*" {
        resultat = firstNum*SecondNum
    }
    if operation == "/" {
        resultat = firstNum/SecondNum
    }
    affichage += "\(resultat)"

}
```

Pour le bouton égal, on ajoute la réserve dans une variable INT dédiée (pour avoir deux valeurs), et avec une condition if déterminant quel signe d'opérateur il s'agit il effectue le calcul puis ajoute le résultat a l'affichage.

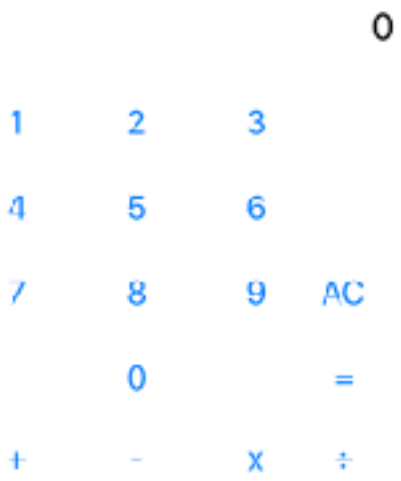
```
labelResult.text = "\(affichage)"
```

Bien sur il ne faut pas oublier d'afficher la variable d'affichage sur le label pour voir l'opération et le résultat.

---

3. Test de la calculatrice

Je teste donc la calculatrice si elle fonctionne, l'opération fonctionne donc bien 24x12 font bien 288.



Et lorsque j'appuie sur le bouton AC l'affichage affiche 0 et réinitialise le tout.

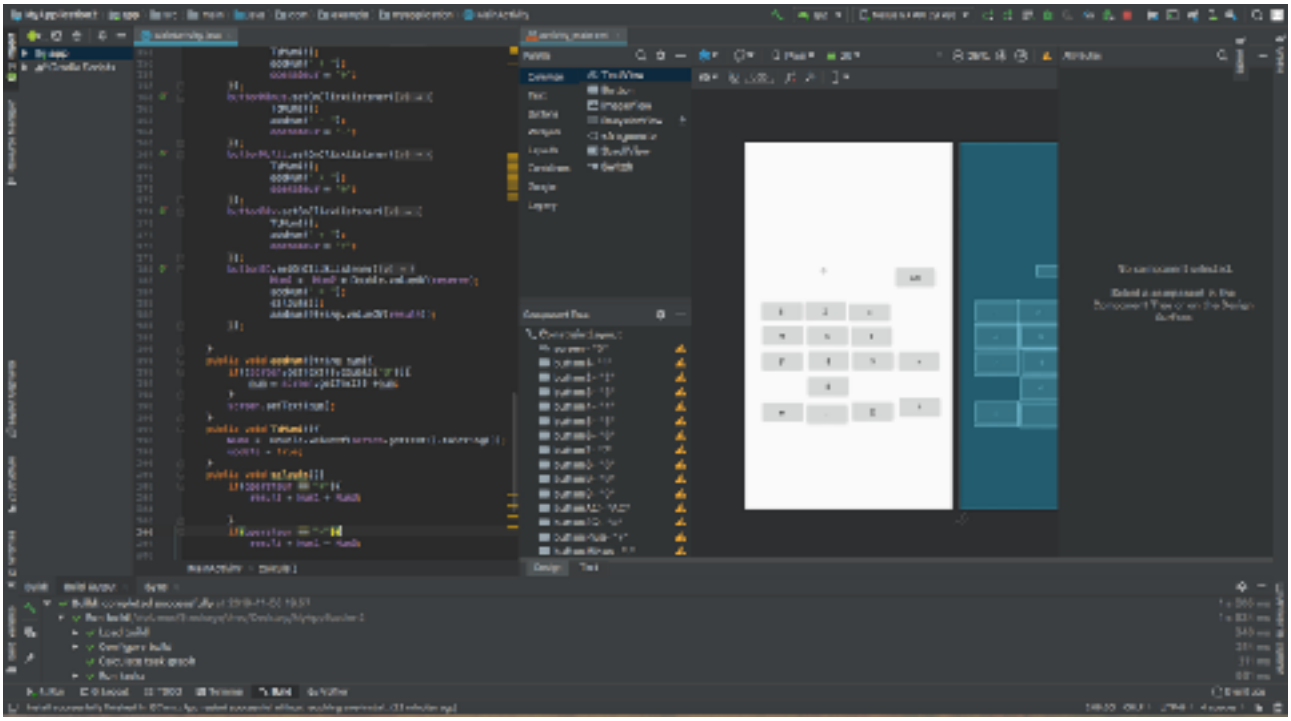
Je passe donc au développement Android.

III. ANDROID STUDIO/JAVA

Android Studio une application détenue par google, est l'équivalent de Xcode mais pour Android, il peut developper des application pour Android tablettes, téléphones, montres connectés, etc..

Pour le langage de programmation j'ai eu le choix entre Kotlin et Java, ayant déjà développé en Java l'année dernière il m'était naturel de choisir Java.

1. Découverte de Android Studio



L'interface de Android Studio n'est pas si différente de celle de Xcode (j'ai aussi « splité » l'interface pour plus de confort), sur le contrôleur en mode design (le contrôleur peux être mis aussi en mode texte) elle présente un menu pour les objets a placer, un pour les objets placé et un menu sur la droite pour configurer les objets.

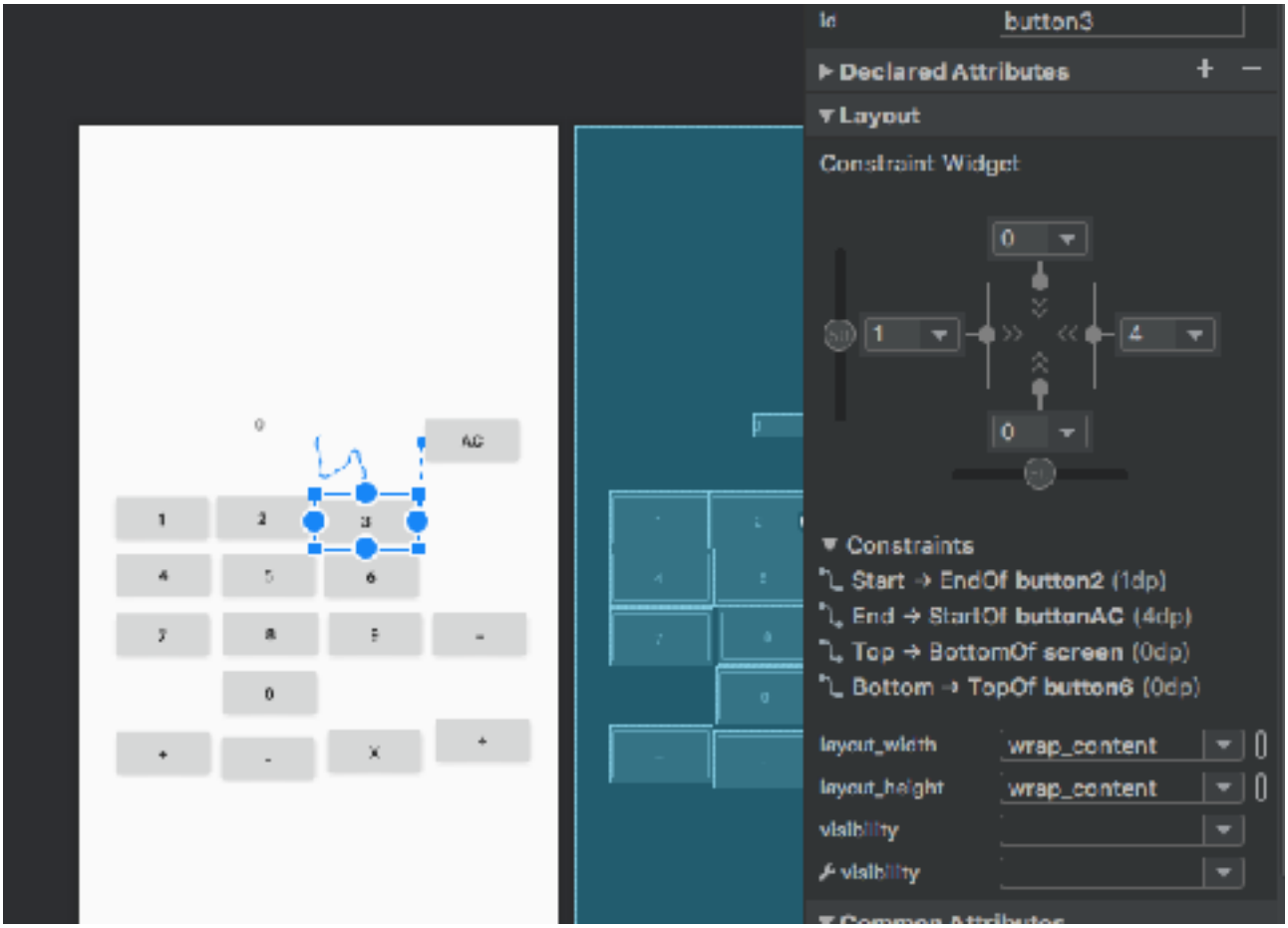
Nous somme muni aussi d'un fichier main, et un option pour simuler un téléphone pour y tester l'application.

```
buttonEQ = (Button)findViewById(R.id.buttonEQ);
screen = (TextView) findViewById(R.id.screen);
```

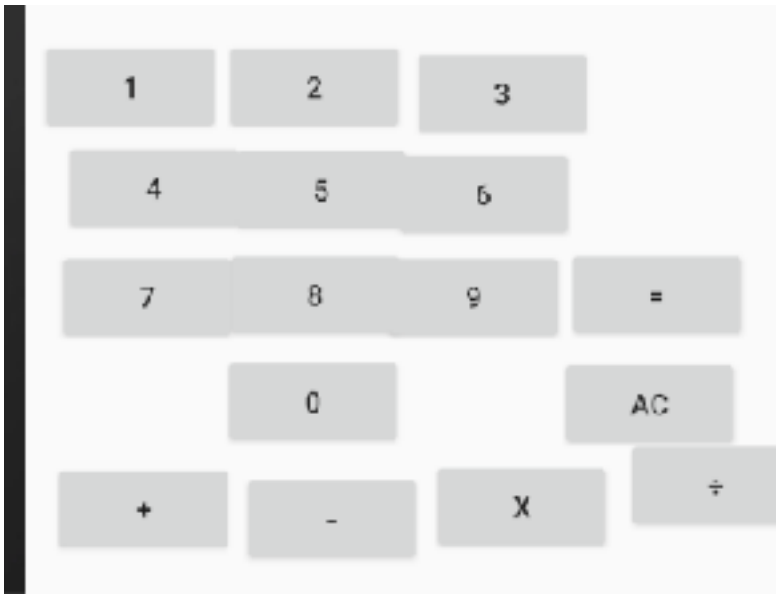
Les liaisons ne ce sont pas vraiment pareil, elle se font dans le code avec une variable comme Xcode, mais ici on ne peux pas « tisser de lien » graphiquement entre l'interface graphique et le code.

```
button1.setOnClickListener((v) → {
```

Il y a des fonction pour les action aussi, également présent sur Xcode.



Toutefois, la fonctionnalité de placement des objets, n'est pas aussi ergonomique que Xcode, car j'ai eu beaucoup de difficultés a placer les boutons correctement.

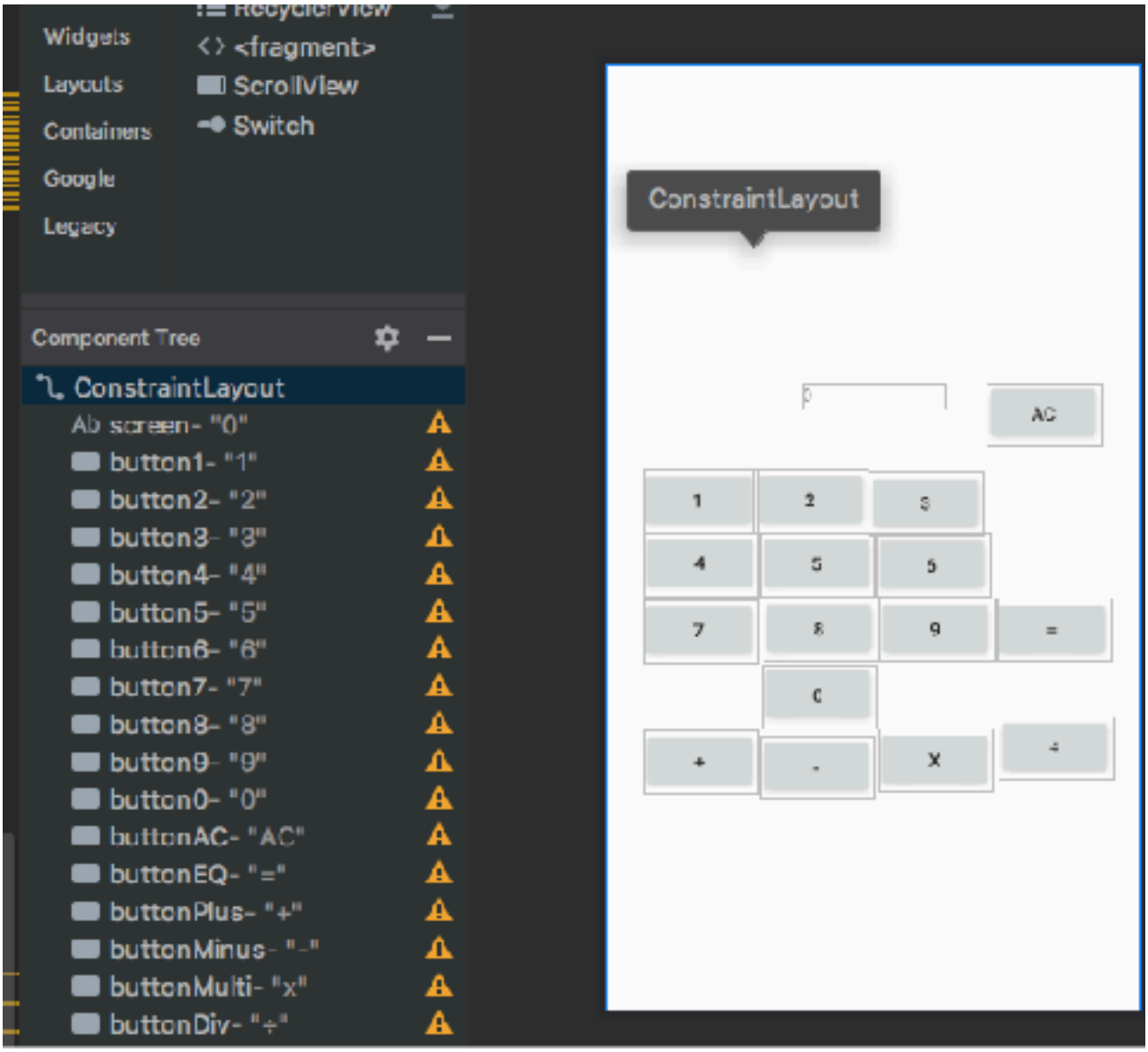


Et lorsque l'application est lancée, les boutons sont encore moins placé comme prévu…

---

2. Développement de la calculatrice

Ayant déjà codé une calculatrice sur Xcode, j'ai donc essentiellement reproduit cette dernière sur Android Studio. Donc cette calculatrice utilisera plus ou moins le même système que la précédente sur Xcode.



J'ai donc tout d'abord placé les boutons de la calculatrice, en leur donnant tous un identifiant.

```
button1 = (Button)findViewById(R.id.button1);
button2 = (Button)findViewById(R.id.button2);
button3 = (Button)findViewById(R.id.button3);
button4 = (Button)findViewById(R.id.button4);
button5 = (Button)findViewById(R.id.button5);
button6 = (Button)findViewById(R.id.button6);
button7 = (Button)findViewById(R.id.button7);
button8 = (Button)findViewById(R.id.button8);
button9 = (Button)findViewById(R.id.button9);
button0 = (Button)findViewById(R.id.button0);
buttonAC = (Button)findViewById(R.id.buttonAC);
buttonPlus = (Button)findViewById(R.id.buttonPlus);
buttonMinus = (Button)findViewById(R.id.buttonMinus);
buttonMulti = (Button)findViewById(R.id.buttonMulti);
buttonDiv = (Button)findViewById(R.id.buttonDiv);
buttonEQ = (Button)findViewById(R.id.buttonEQ);
screen = (TextView) findViewById(R.id.screen);
```

Puis je les ai intégré dans le code dans des variables.

```
private String operateur = "";
private double Num1;
private double Num2;
private double result;
private String reserve = "";
private boolean update = false;
```

J’ai ensuite crée les variable nécessaire a la calculatrice (j’ai rajouté une variable booléenne pour savoir si le premier numéro est passé pour passer au suivant après l’opérateur, un système different de celui utilisé sur la calculatrice XCode).

```
button1.setOnClickListener((v) → {
        addNum("1");
        if(update){
            reserve = 1 + reserve;
        }
});
```

J’ai donc codé par la suite les actions des boutons chiffres qui permettent avec la method addNum d’ajouter le chiffre sur l’affichage de la calculatrice.

```
public void addNum(String num){
    if(!screen.getText().equals("0")){
        num = screen.getText() +num;
    }
    screen.setText(num);
}
```

```
public void calculs(){
    if(operateur == "+"){
        result = Num1 + Num2;

    }
    if(operateur == "-"){
        result = Num1 - Num2;

    }
    if(operateur == "*"){
        result = Num1 * Num2;

    }
    if(operateur == "/"){
        result = Num1 / Num2;

    }
}
```

J'ai ajouté, les fonctions des actions des opérateurs (gardant de côté le signe dans une variable) et la méthode qui calculera le résultat.

```
buttonEQ.setOnClickListener((v) → {
        Num2 =  Num2 + Double.valueOf(reserve);
        addNum(" = ");
        calculs();
        addNum(String.valueOf(result));
});
```

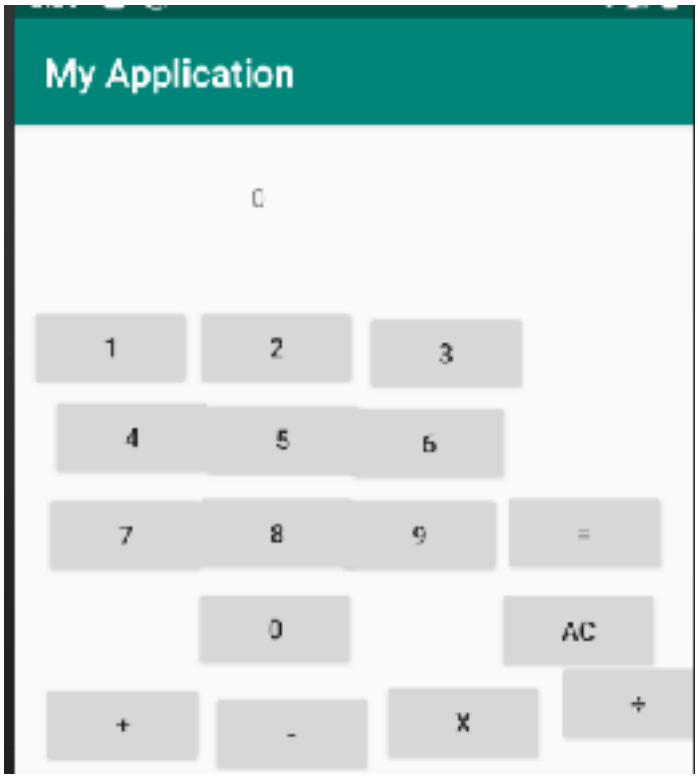
Puis le bouton pour l'afficher.

```
});
buttonAC.setOnClickListener((v) → {
        screen.setText("0");
        update = false;
        reserve = ("");
        Num1 = 0;
        Num2 = 0;
        operateur = "";
        result = 0;
});
```

Et celui pour tout réinitialiser.

3. Test de la calculatrice

J’ai donc essayé l’application, et 12x2 font bien 24.



J’appuie sur le bouton AC et tout ce réinitialise, l’affichage affiche donc 0.