

# Storyboards et UIKit

Votre UI part de là...

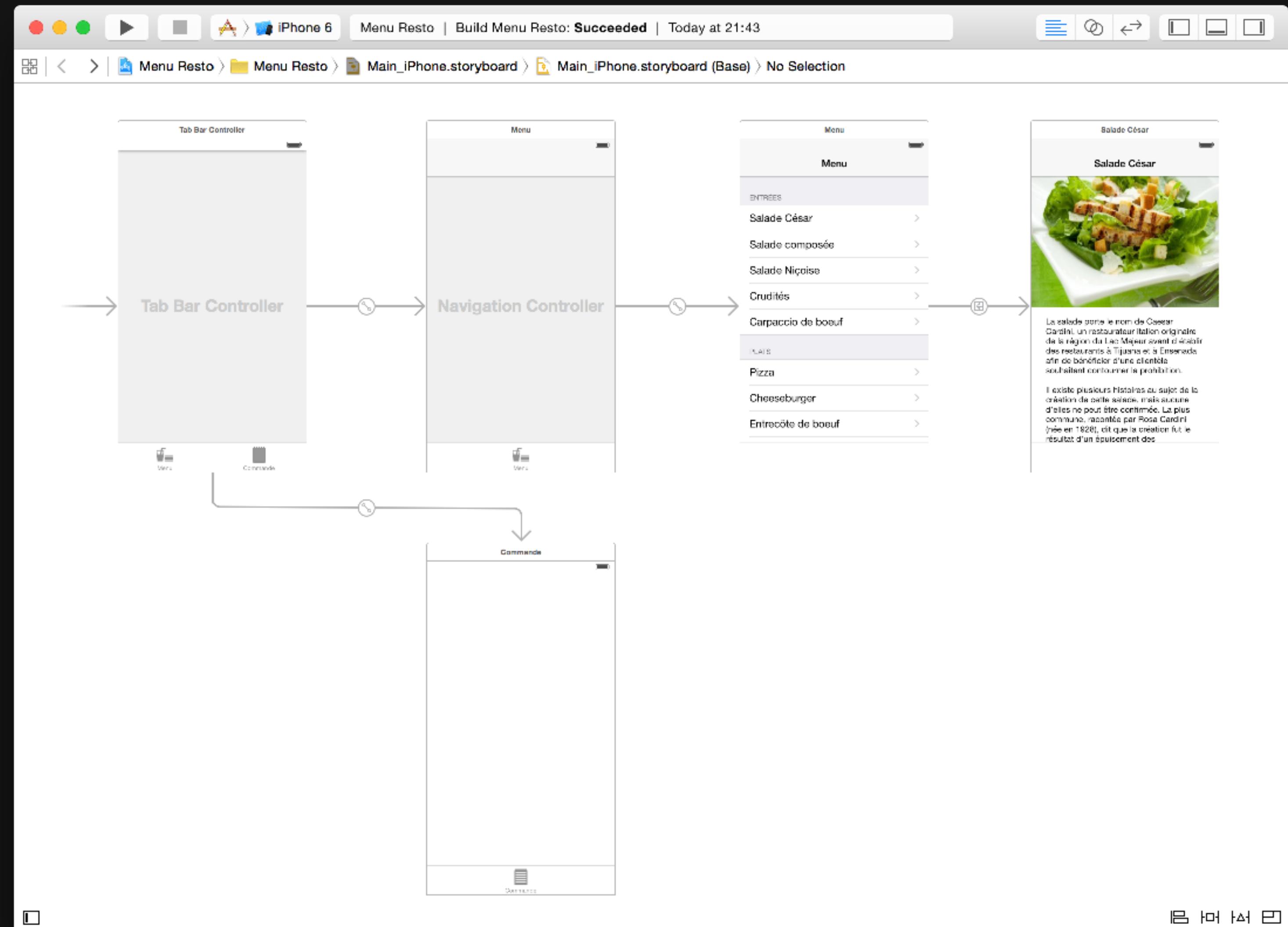
# Storyboards et UIKit

- Storyboards
- UIKit : organisation
- Adaptabilité de l'interface
- Objets courants

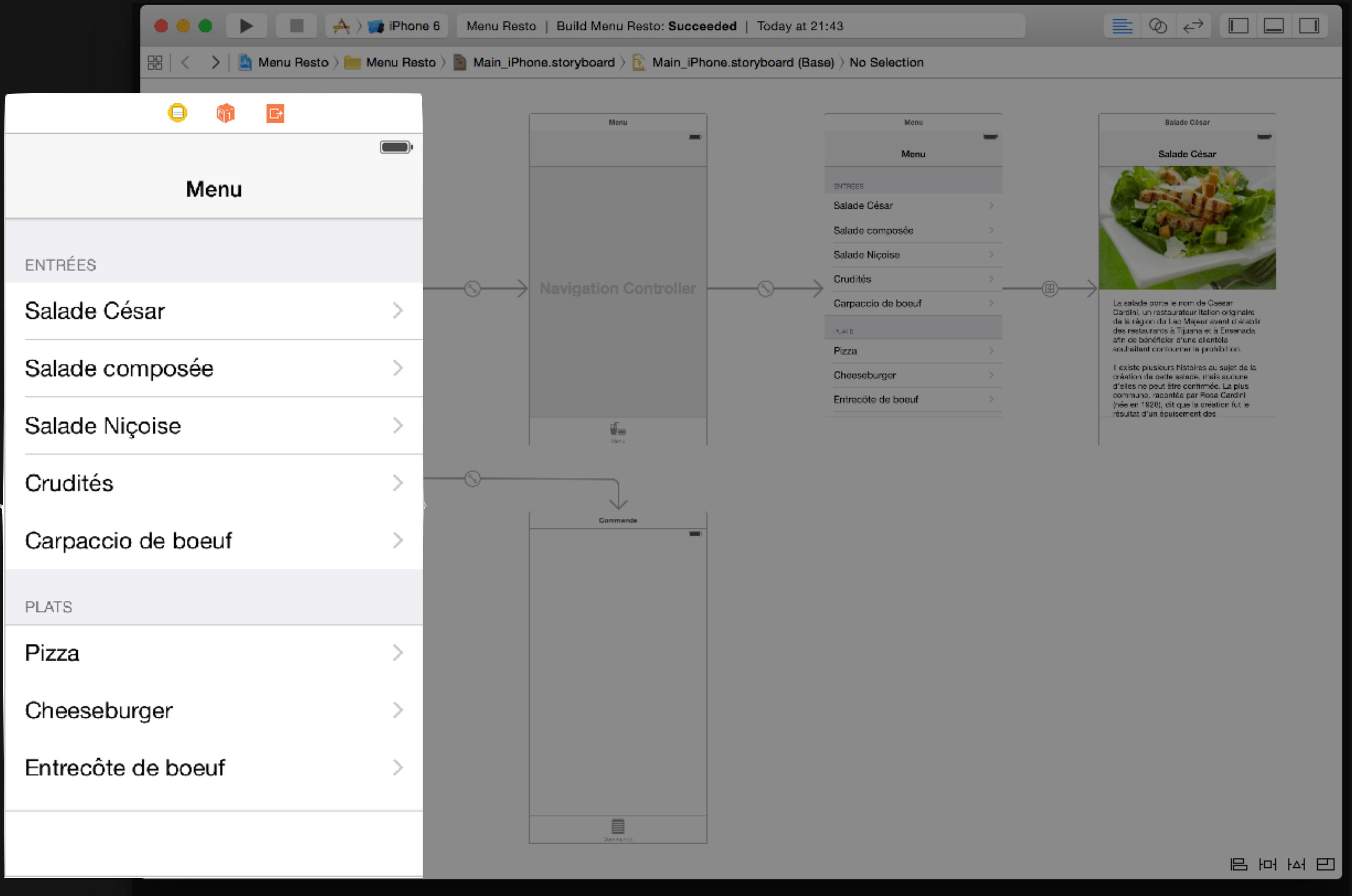
# Storyboards

- Façon de concevoir son UI
- Fonctionne en reliant des scènes via des segues
- Chaque scène représente une instance de UIViewController
- Un UIViewController présente une vue, ou parfois encapsule un autre UIViewController

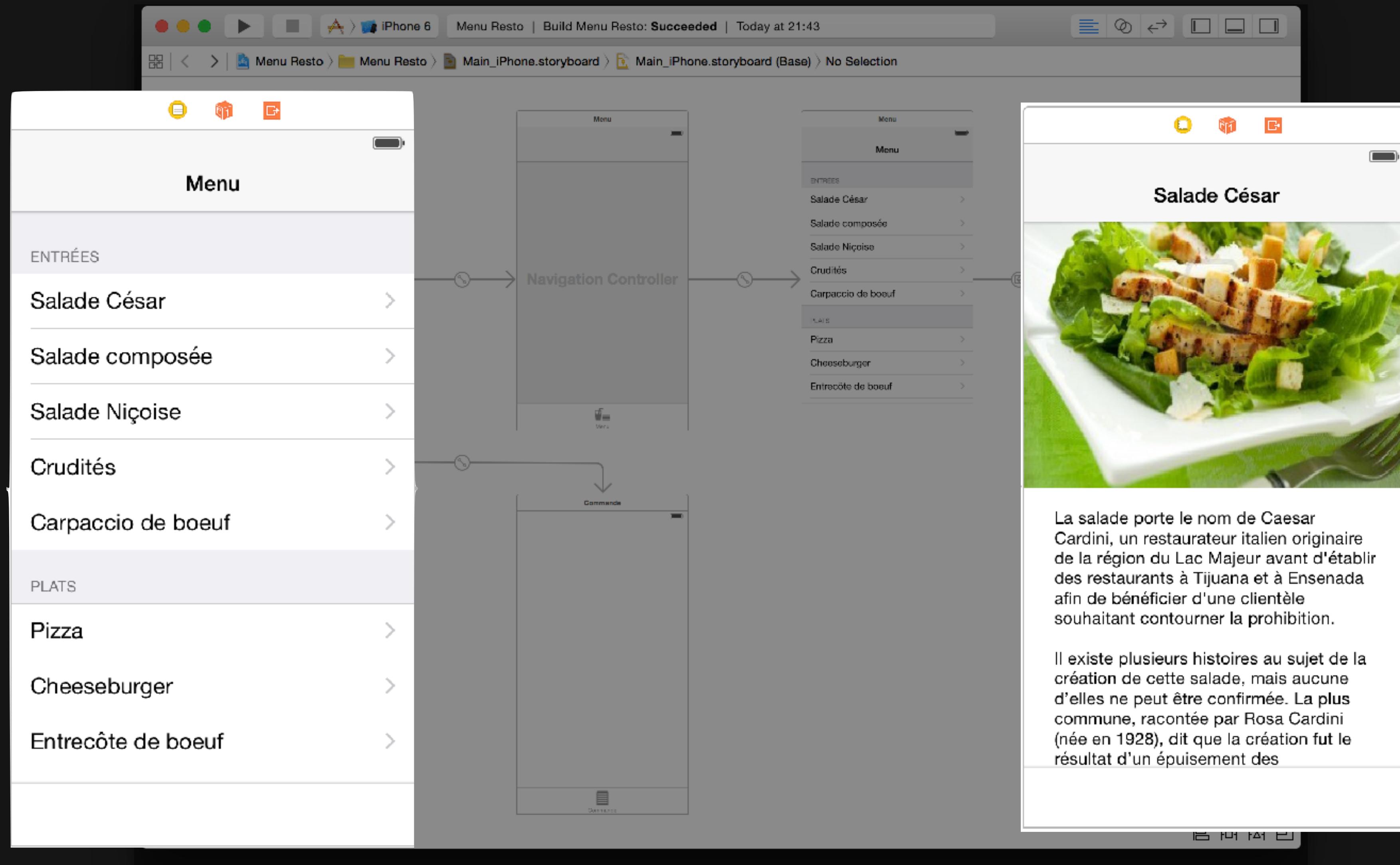
# Storyboards



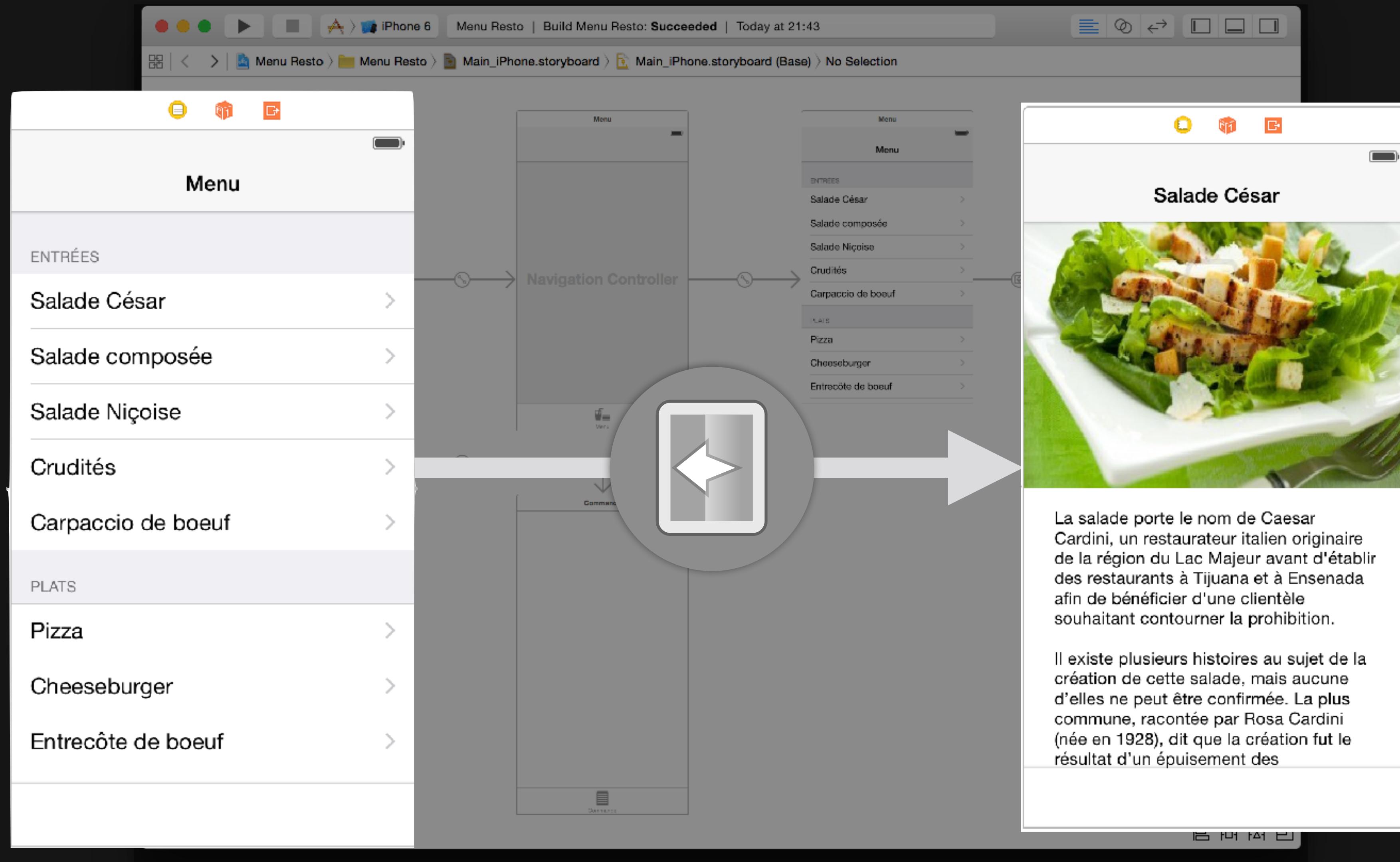
# Storyboards



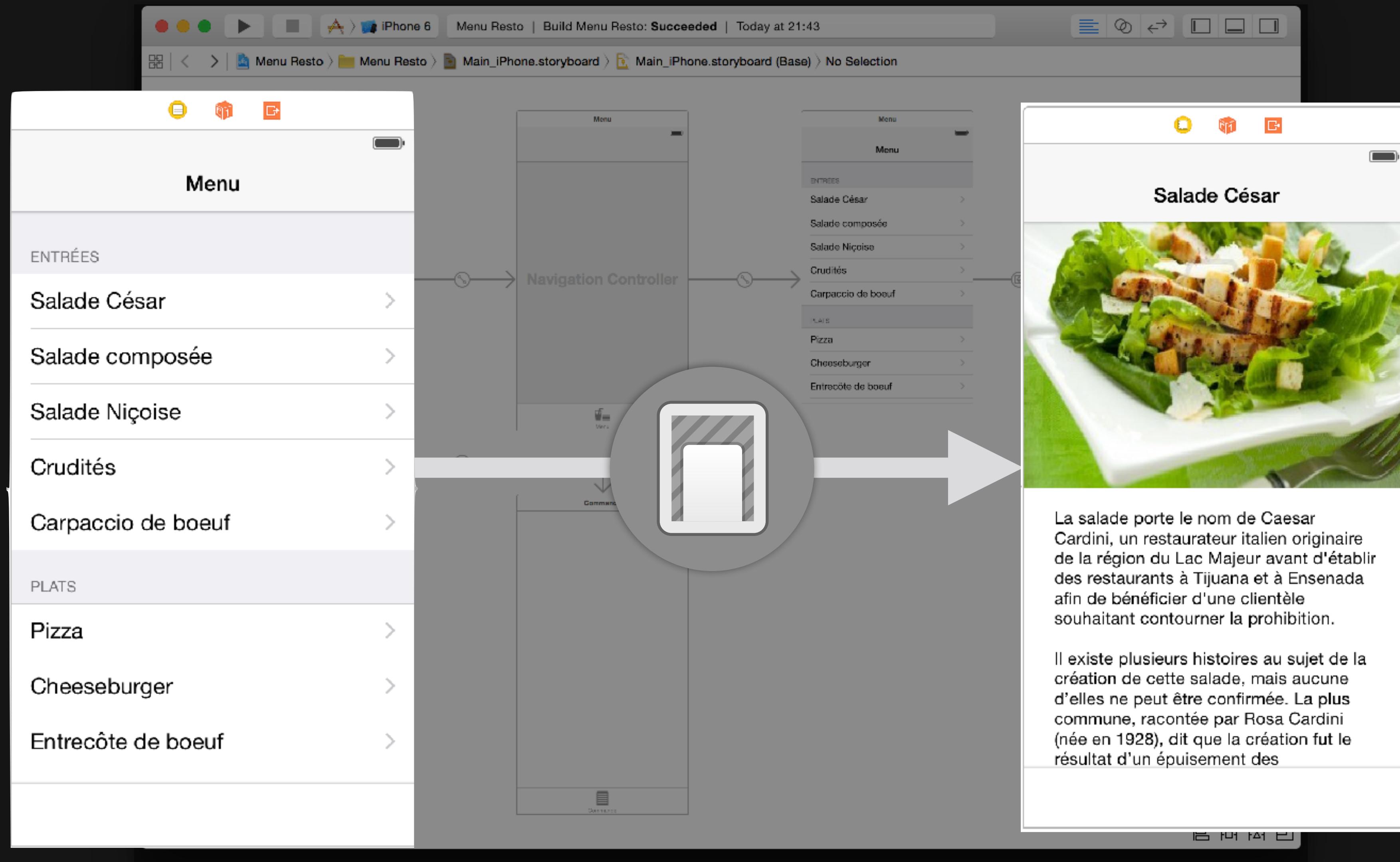
# Storyboards



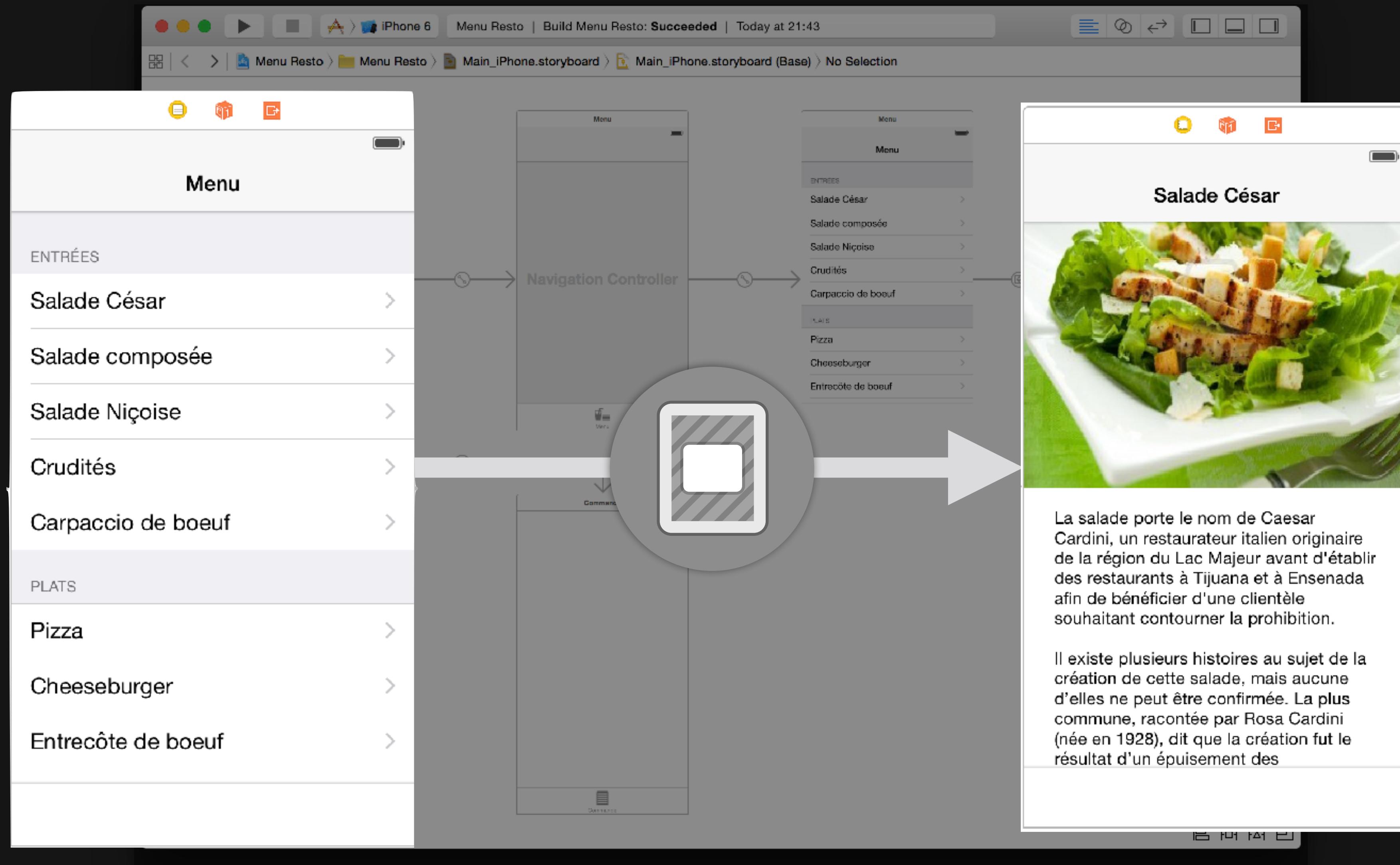
# Storyboards



# Storyboards



# Storyboards



# Storyboards



Carrier ⌘ 10:55 PM

## Menu

ENTRÉES

- Salade César >
- Salade composée >
- Salade Niçoise >
- Crudités >
- Carpaccio de boeuf >

PLATS

- Pizza >
- Cheeseburger >
- Entrecôte de boeuf >

 Menu

 Commande

Carrier ⌘ 10:55 PM

< Menu Salade César



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épuisement des

 Menu

 Commande



prepare(for segue:, sender:)

Carrier 10:55 PM

Menu

ENTRÉES

- Salade César >
- Salade composée >
- Salade Niçoise >
- Crudités >
- Carpaccio de boeuf >

PLATS

- Pizza >
- Cheeseburger >
- Entrecôte de boeuf >

Menu Commande



Carrier 10:55 PM

< Menu Salade César



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épuisement des

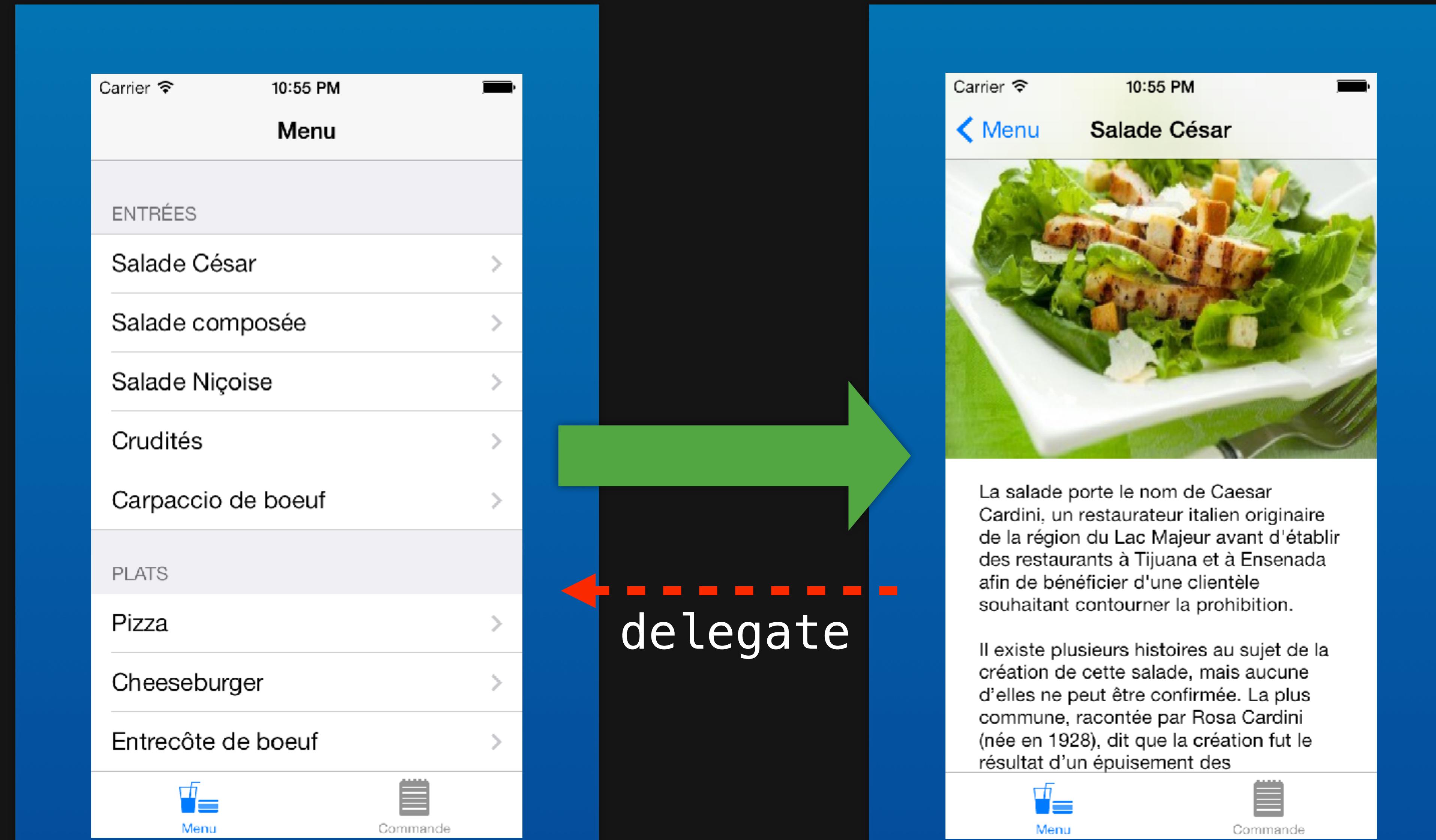
Menu Commande

# Storyboards



```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if segue.identifier == "displayDetails" {  
        var nextViewController = segue.destination  
        nextViewController.title = "The title of the next viewController"  
    }  
}
```

# Storyboards

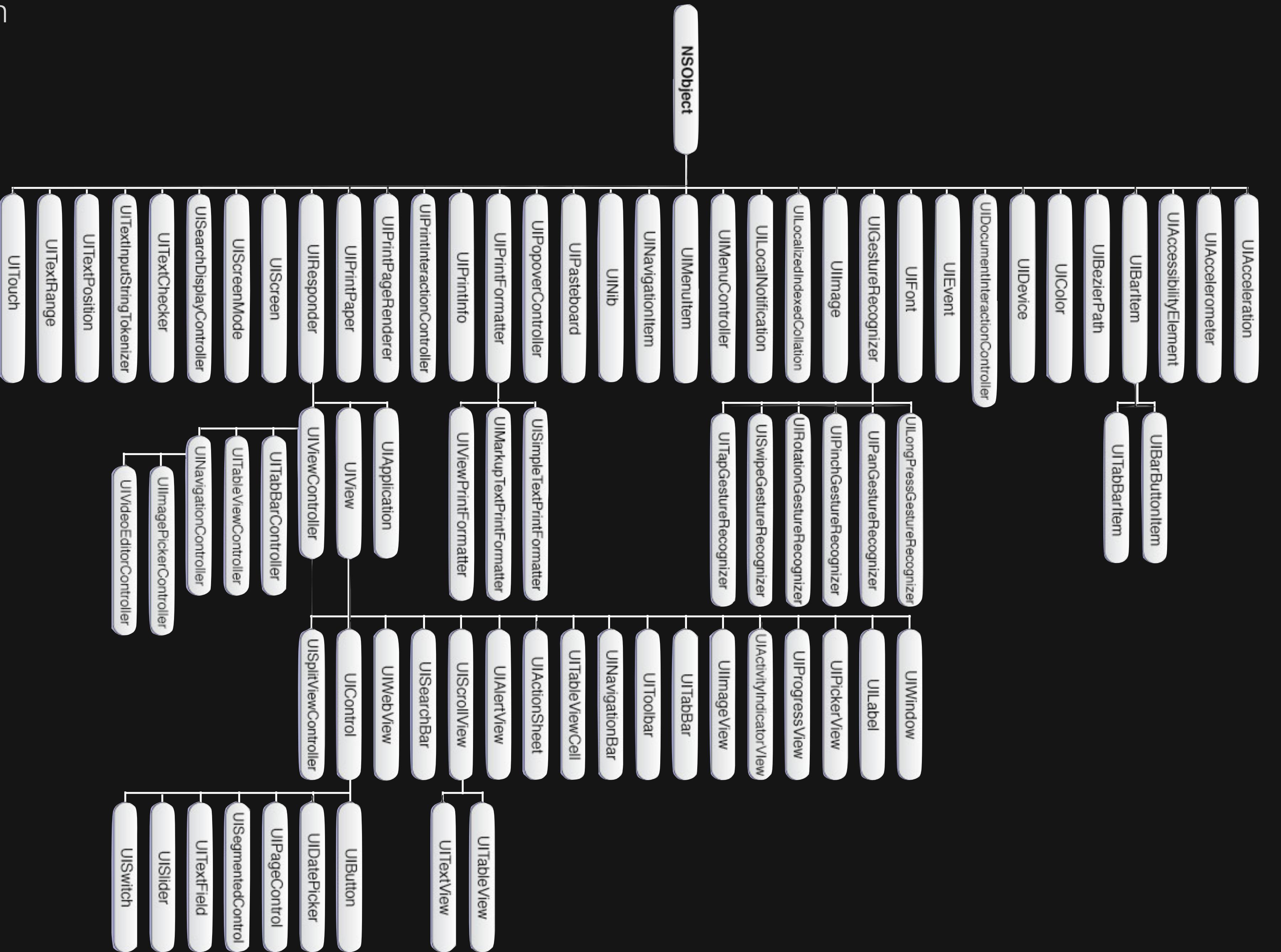


# UIKit : organisation

## UIKit : organisation

- UIKit → Framework
- Contient toutes les classes pour vos UI
  - UIWindow, UIButton, UILabel...
- Travail simplifié pour le développeur

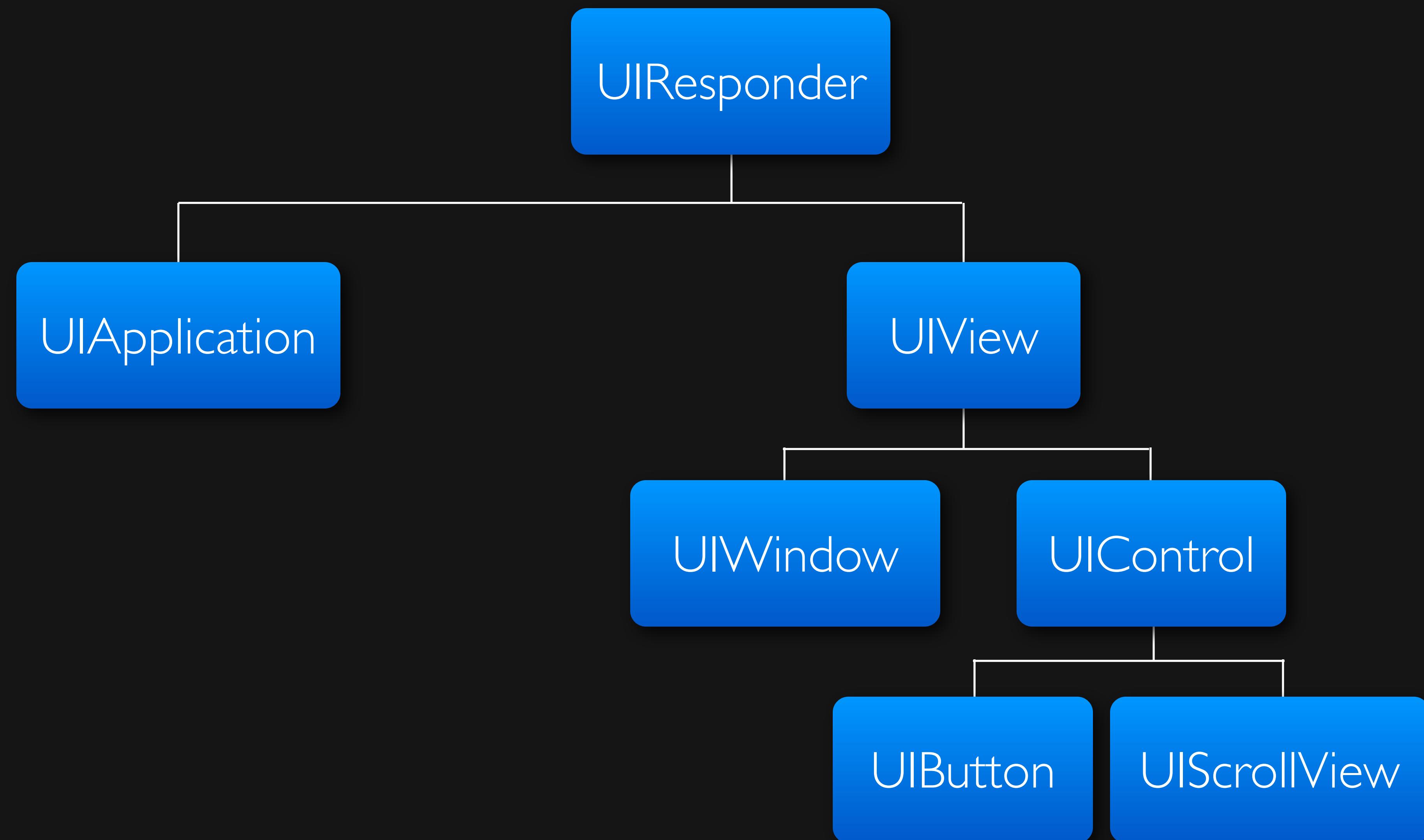
# UIKit : organisation



# UIKit : organisation

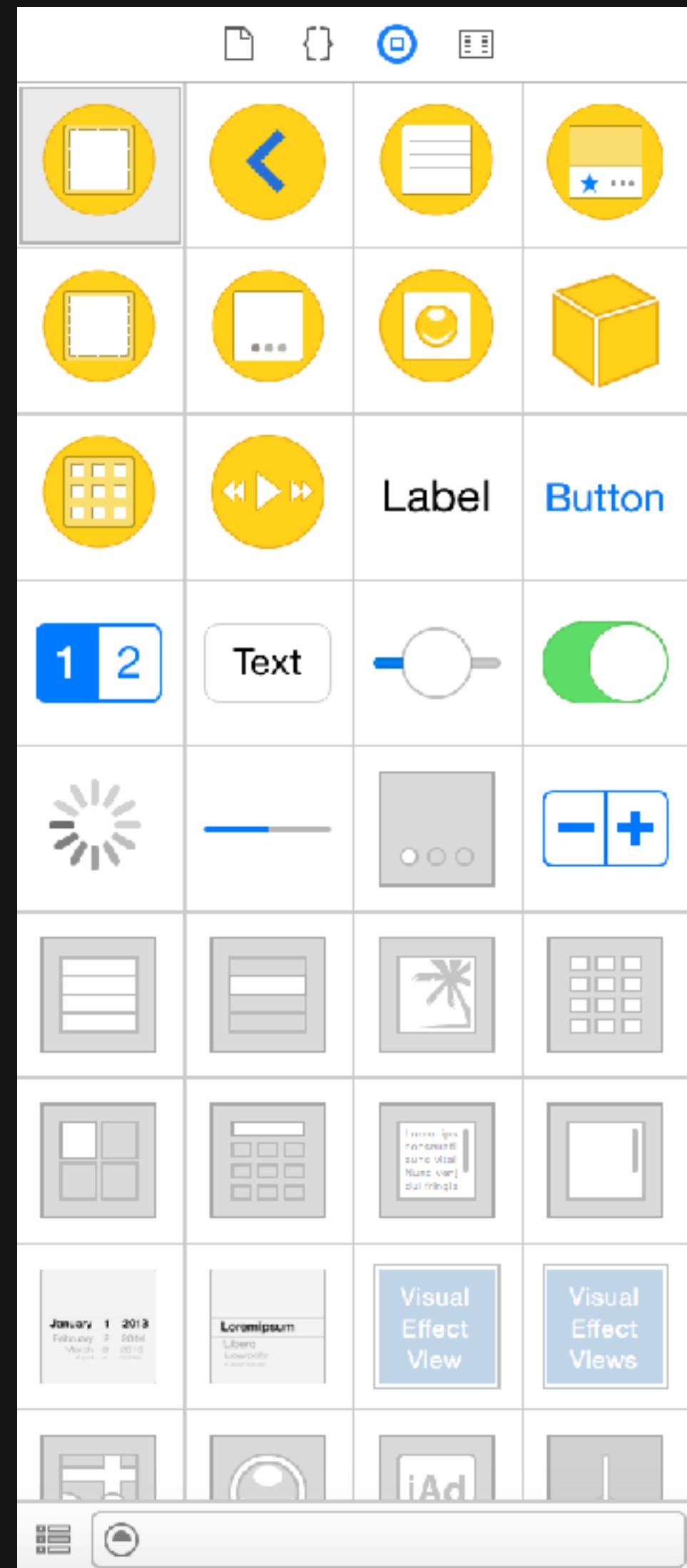


## UIKit : organisation



## UIKit : organisation

- Une grande variété d'objets disponibles
- Accessibles en drag'n drop, ou via le code





- UIResponder
- Gestion des touches
  - `func touchesBegan(_ touches: NSSet, withEvent event: UIEvent)`
  - `func touchesCancelled(_ touches: NSSet!, withEvent event: UIEvent!)`
  - `func touchesEnded(_ touches: NSSet, withEvent event: UIEvent)`
  - `func touchesMoved(_ touches: NSSet, withEvent event: UIEvent)`

# UIWindow

- Point de départ de l'interface
- Une app iOS ne possède, en général qu'une fenêtre



# UIView

- Zone rectangulaire permettant d'afficher du contenu
- Par défaut : un rectangle rempli avec une couleur de fond
- Peut être sous-classé pour être personnalisé
- Une vue peut contenir des sous-vues
  - `var subviews: [UIView] { get }`
- Une vue peut posséder une vue parente
  - `var superview: UIView? { get }`

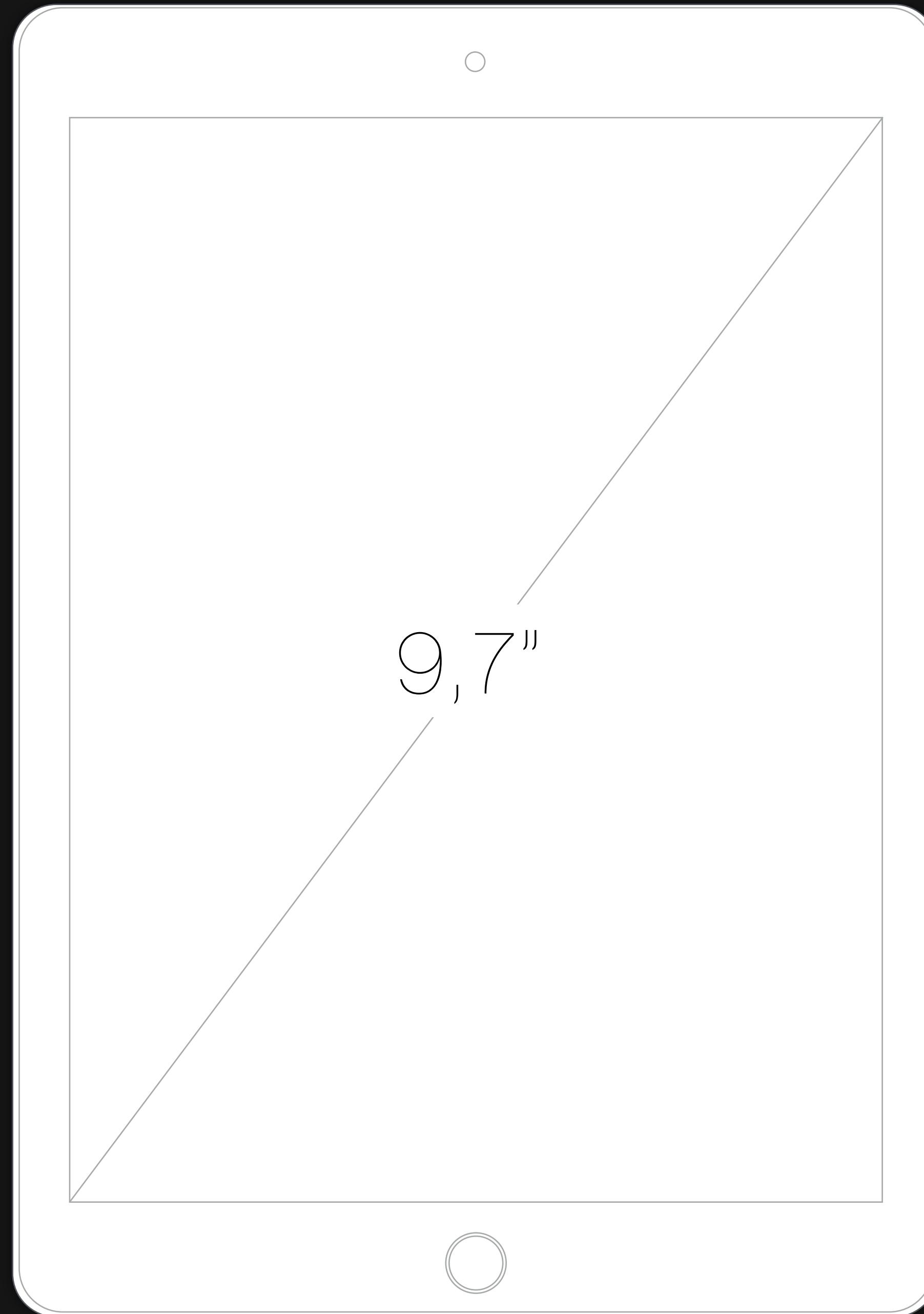
# UIControl

- Sous-classe de UIView
- Permet de réagir simplement aux taps en envoyant des actions (IBAction)

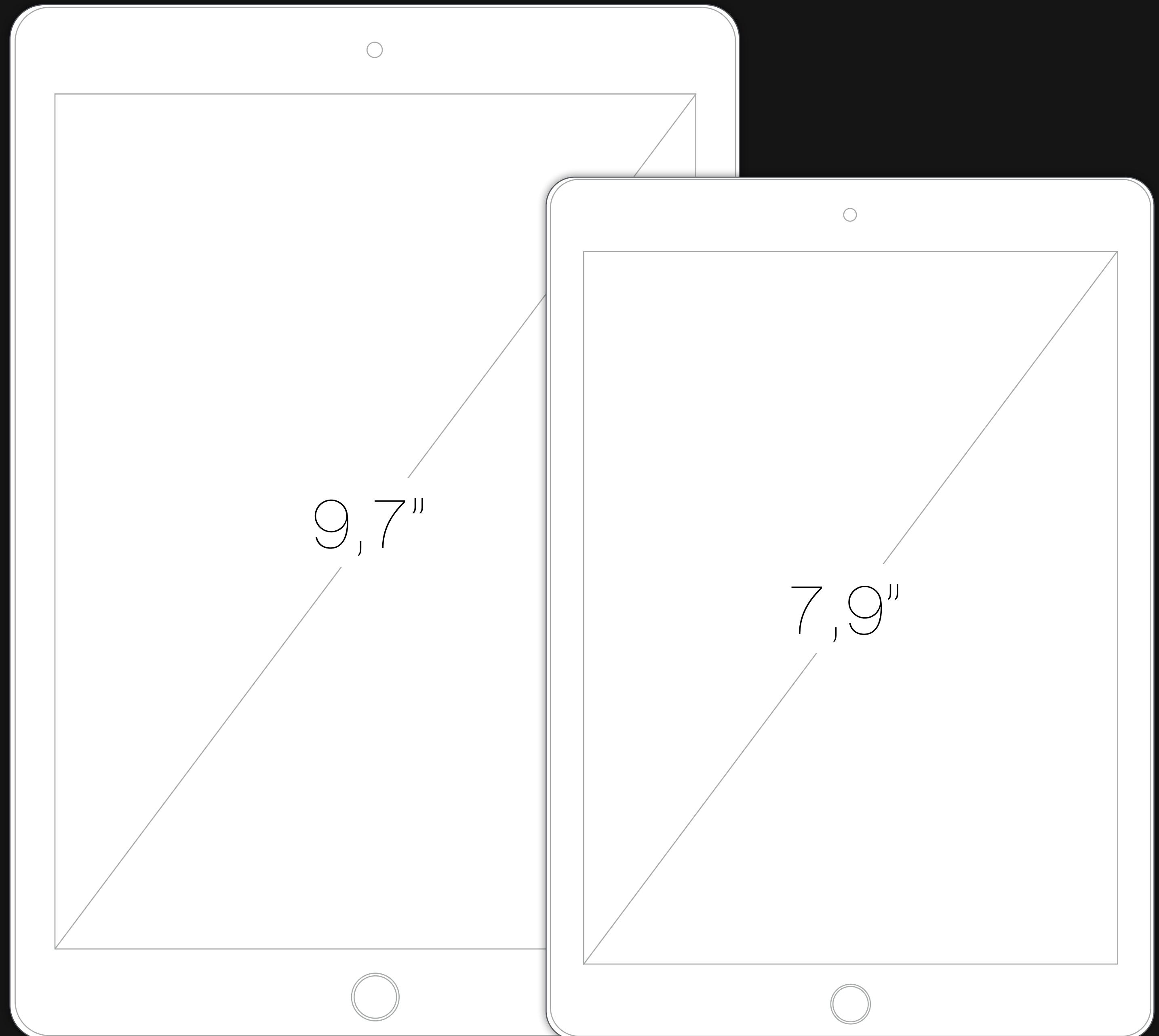
# Adaptabilité de l'interface



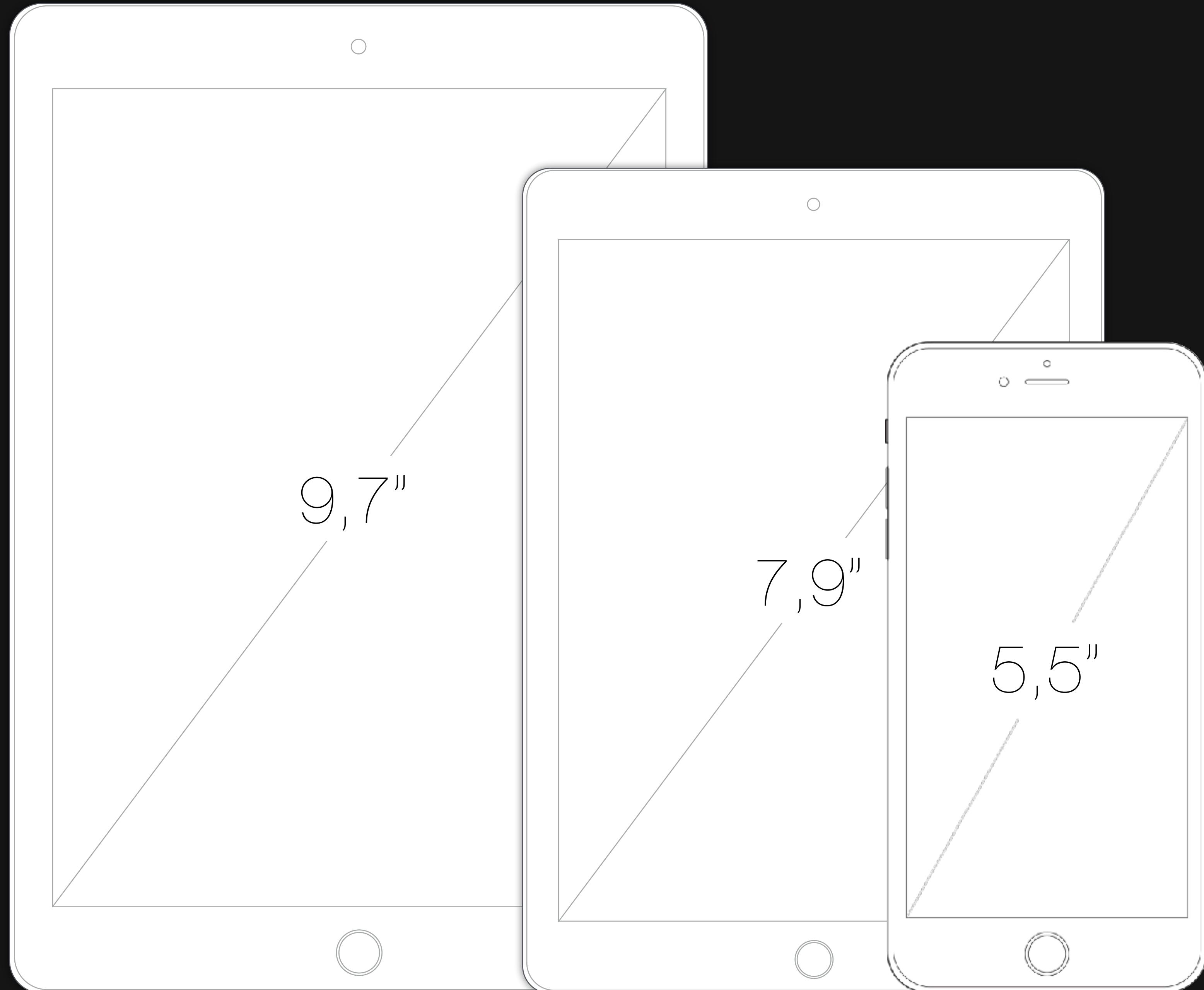
## Adaptabilité de l'interface



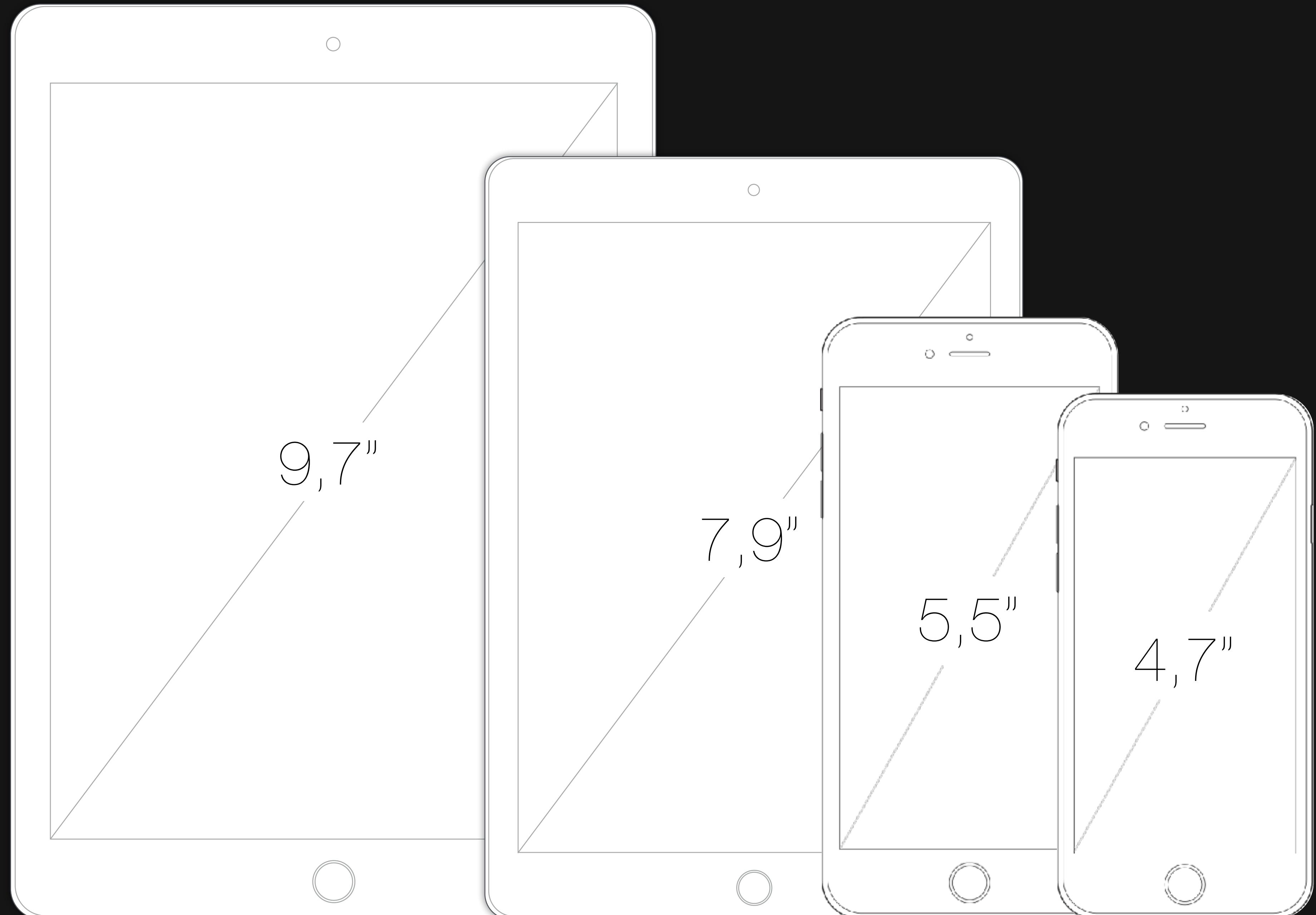
## Adaptabilité de l'interface



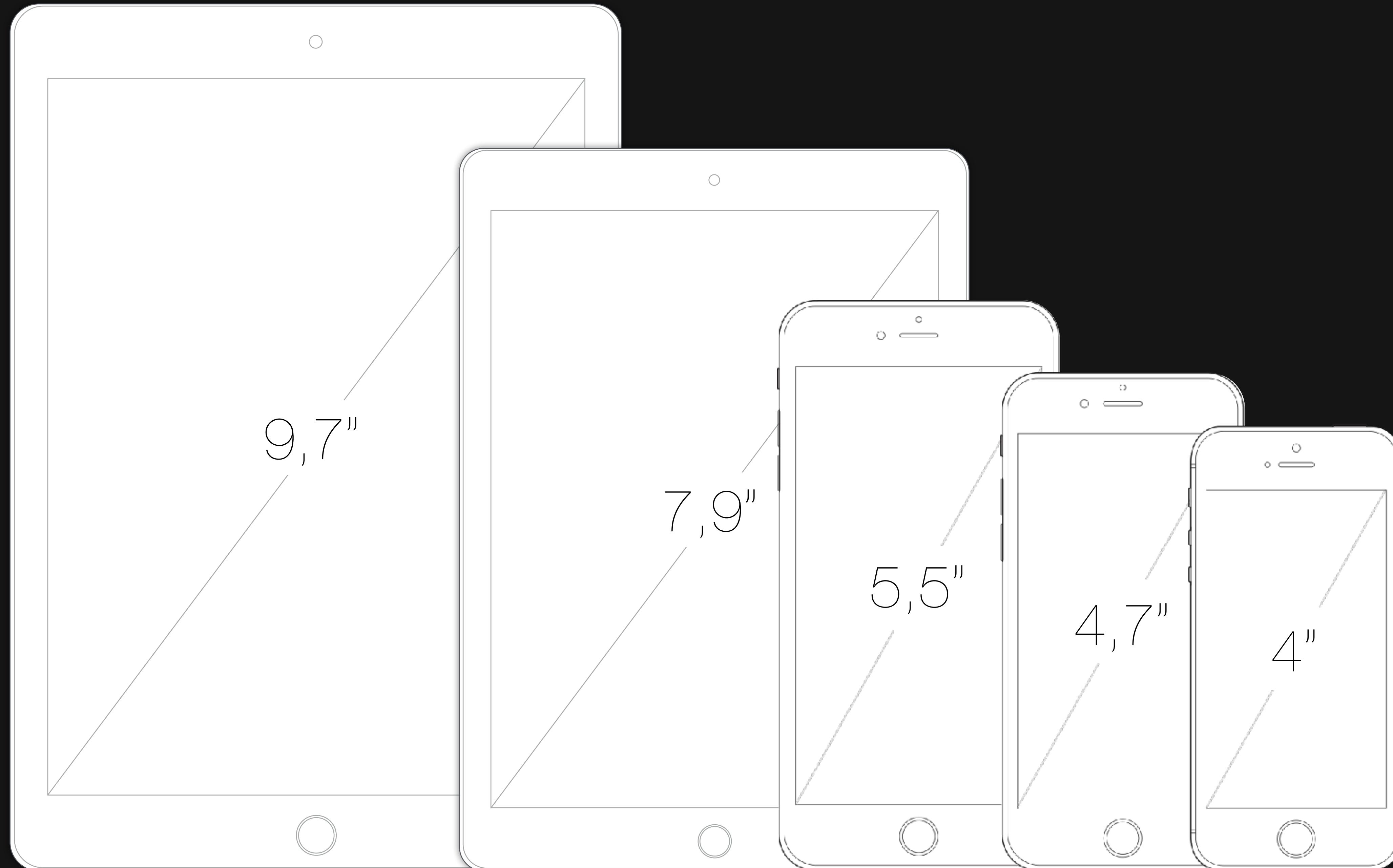
## Adaptabilité de l'interface



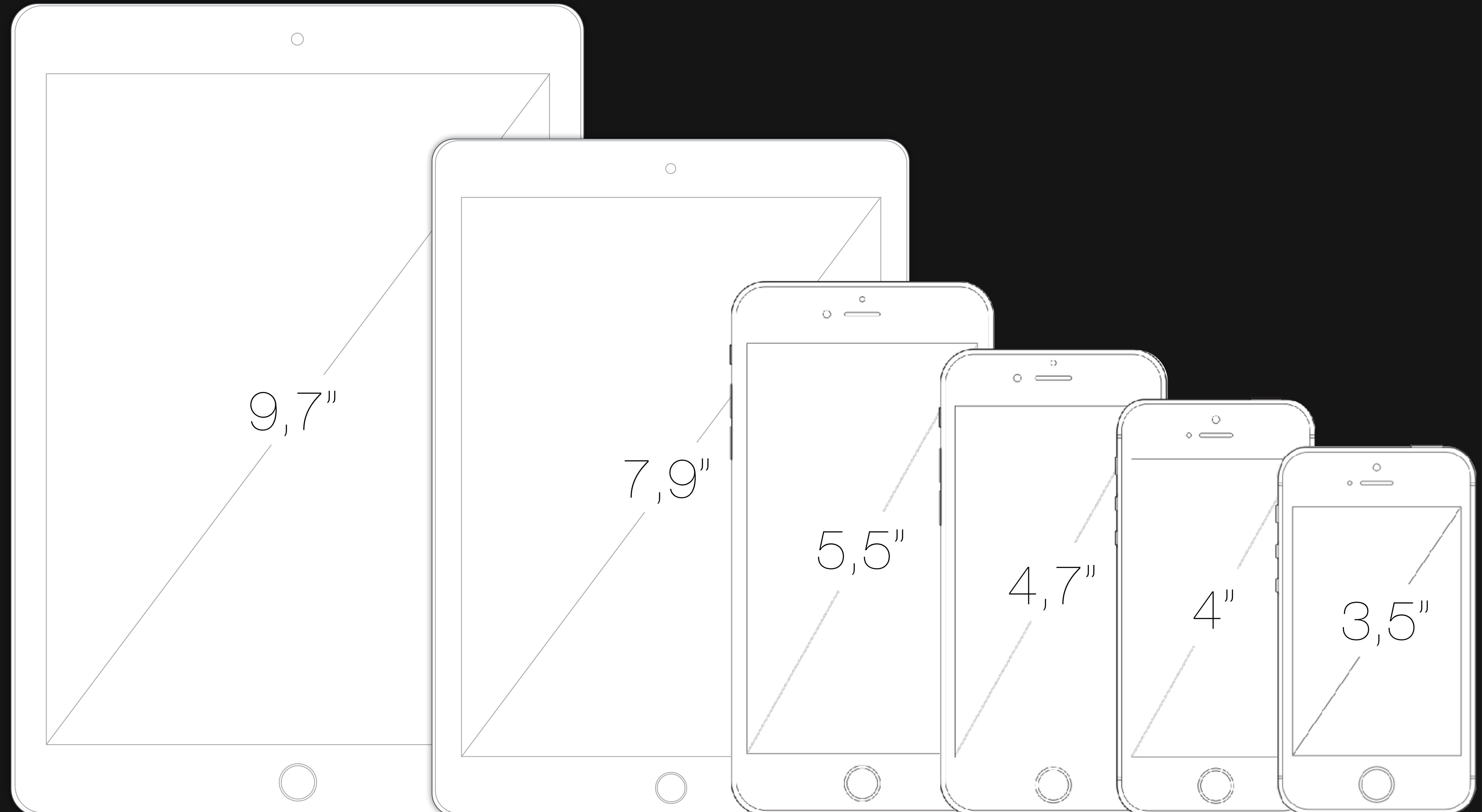
## Adaptabilité de l'interface



## Adaptabilité de l'interface



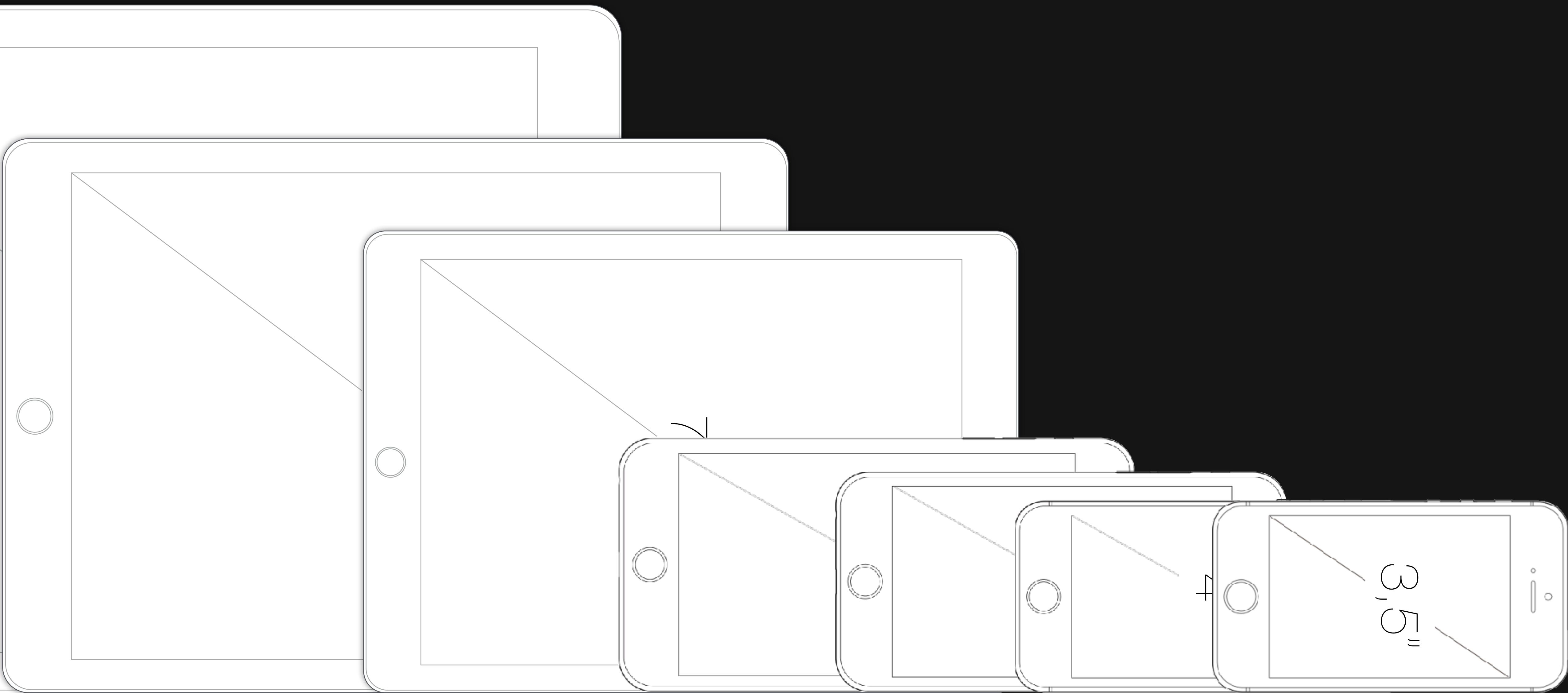
## Adaptabilité de l'interface



## Adaptabilité de l'interface



## Adaptabilité de l'interface



## Adaptabilité de l'interface

- iOS fonctionne sur une grande variété d'appareils
- Chaque appareil peut être utilisé en portrait ou paysage
- Cela donnerait énormément d'écrans à concevoir
- Auto Layout, les classes de tailles et les contrôleurs adaptatifs nous aident à réduire ce travail

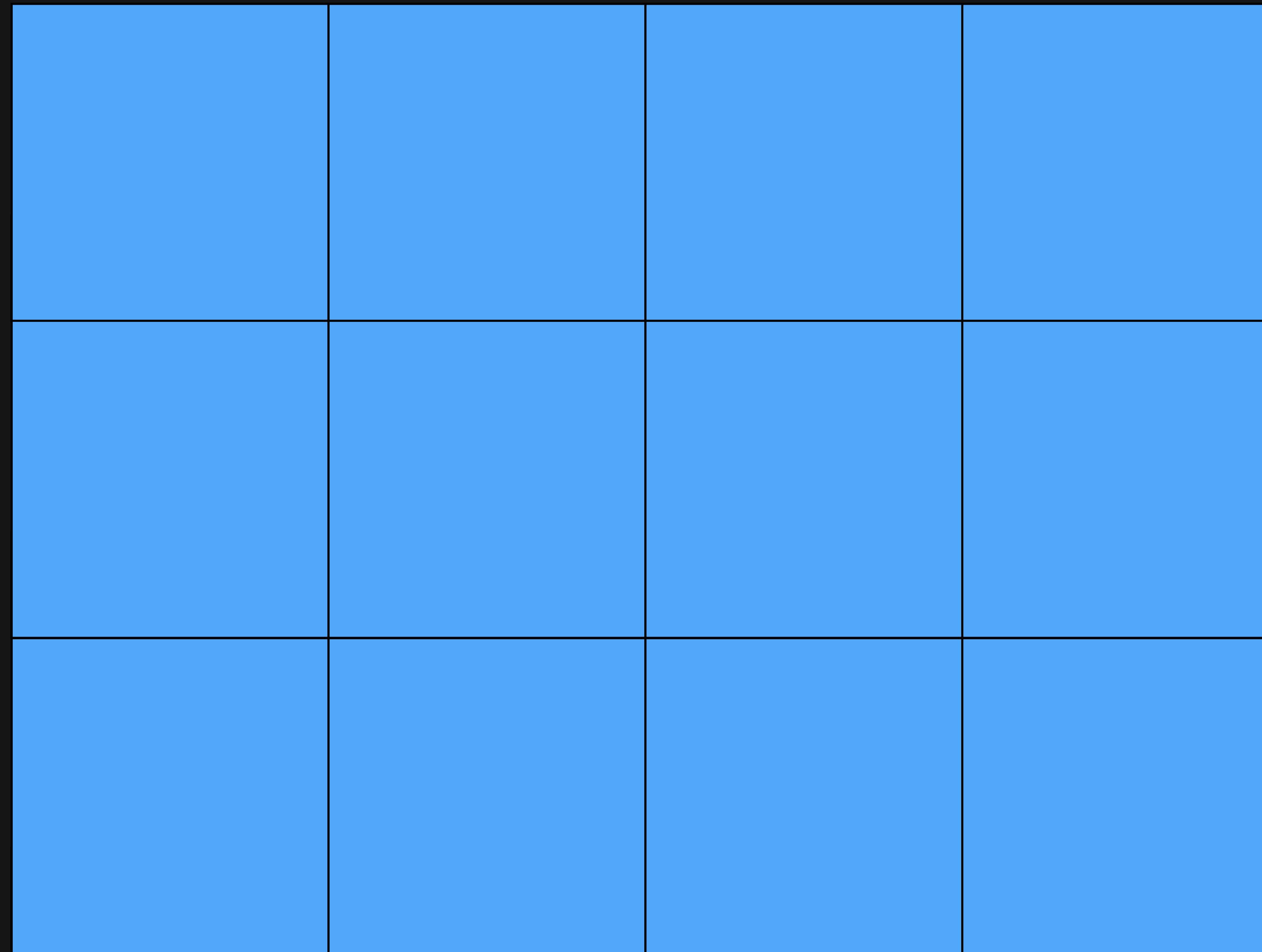
# Coordonnées

- CGFloat : type défini à partir de float ou de double utilisé dans Core Graphics
- CGPoint : structure représentant un point
  - (CGFloat x, CGFloat y)
- CGSize : structure représentant une taille
  - (CGFloat width, CGFloat height)
- CGRect : structure représentant un rectangle
  - (CGPoint origin, CGSize size)

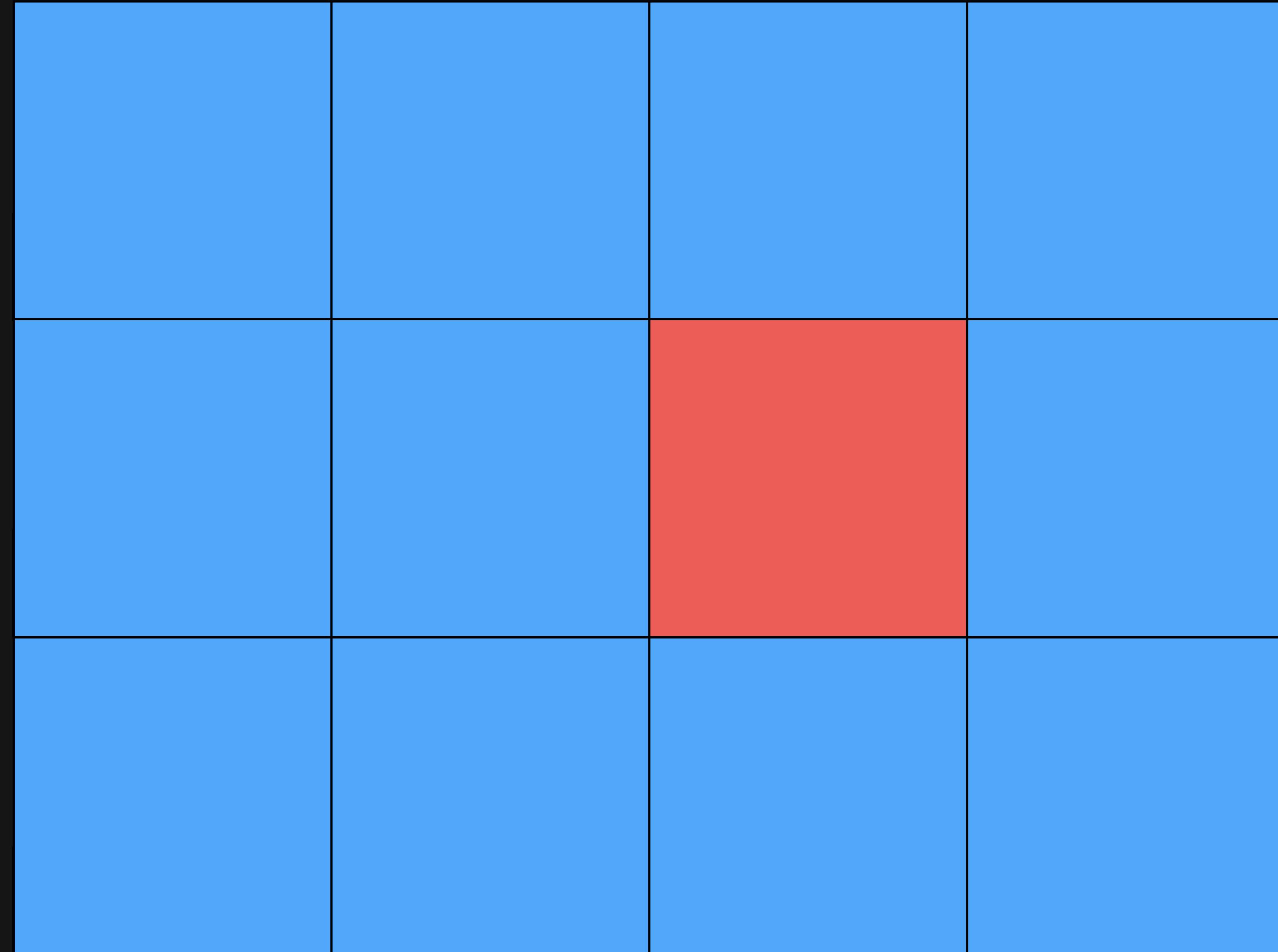
# Coordonnées

- Origine du système de coordonnées en haut à gauche
- Coordonnées en *points*. Les *points* sont liés aux *pixels* en fonction de la densité de l'écran.

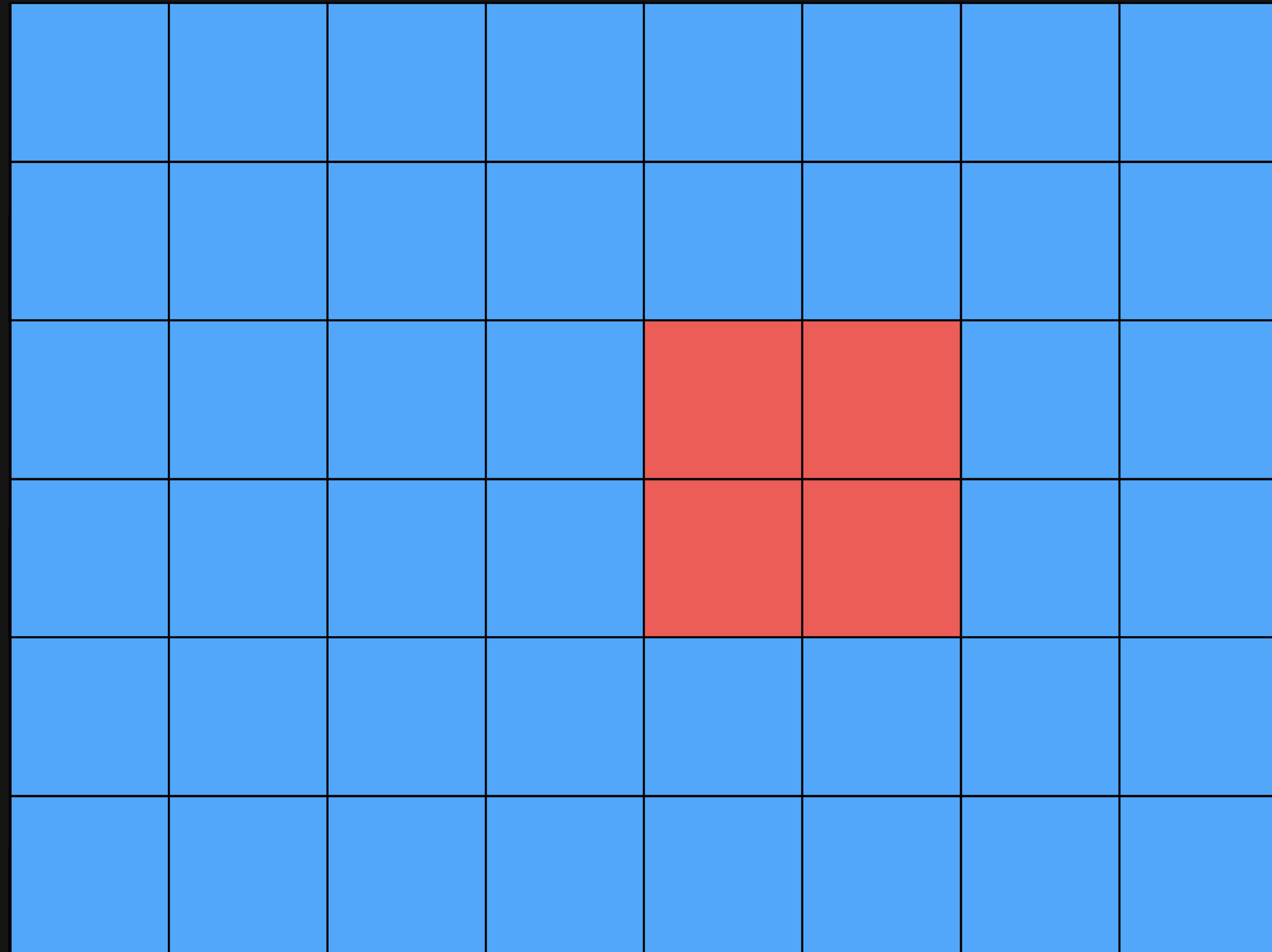
## Adaptabilité de l'interface



## Adaptabilité de l'interface



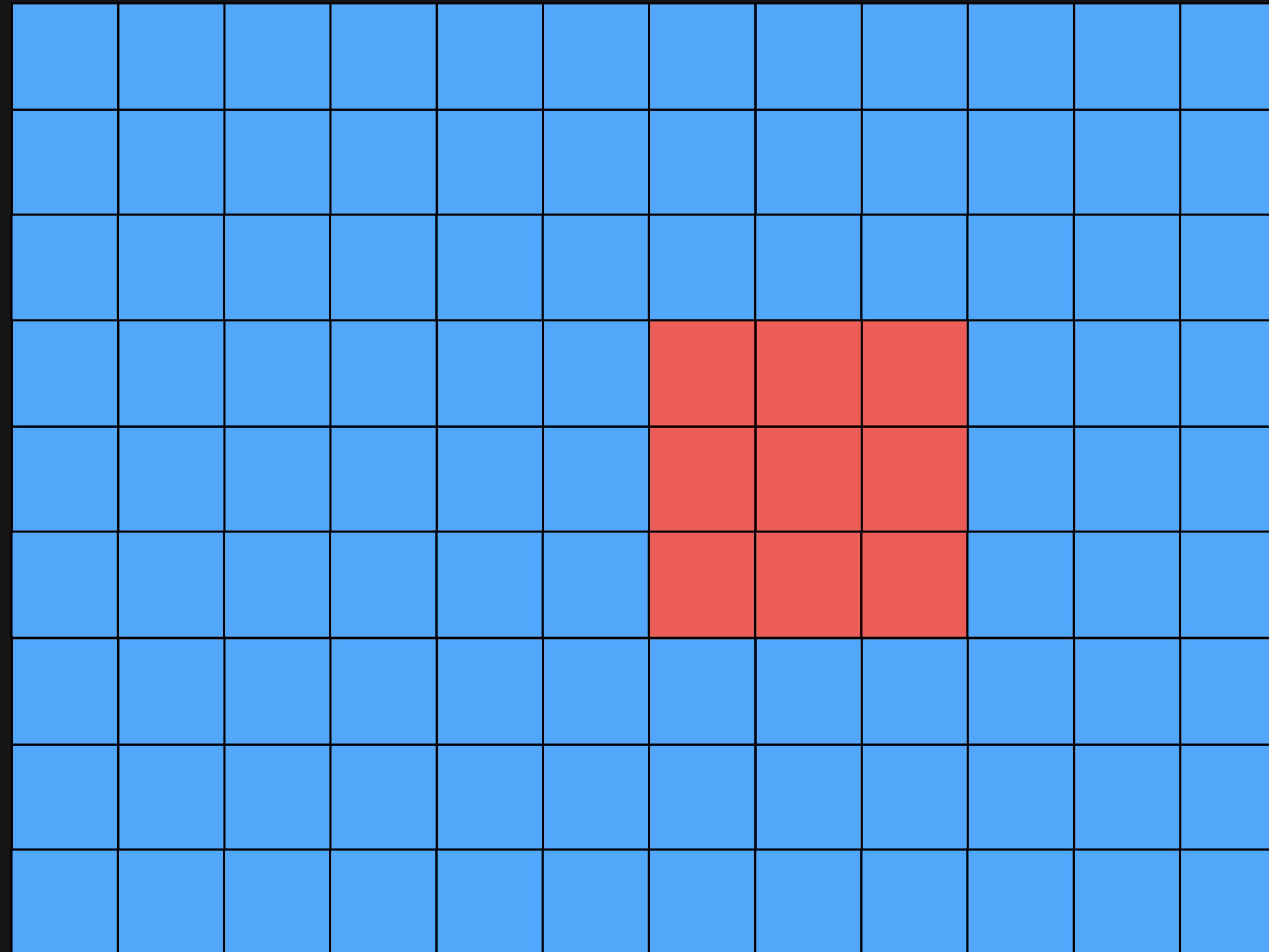
## Adaptabilité de l'interface



Points : (3,2)

@2x

## Adaptabilité de l'interface

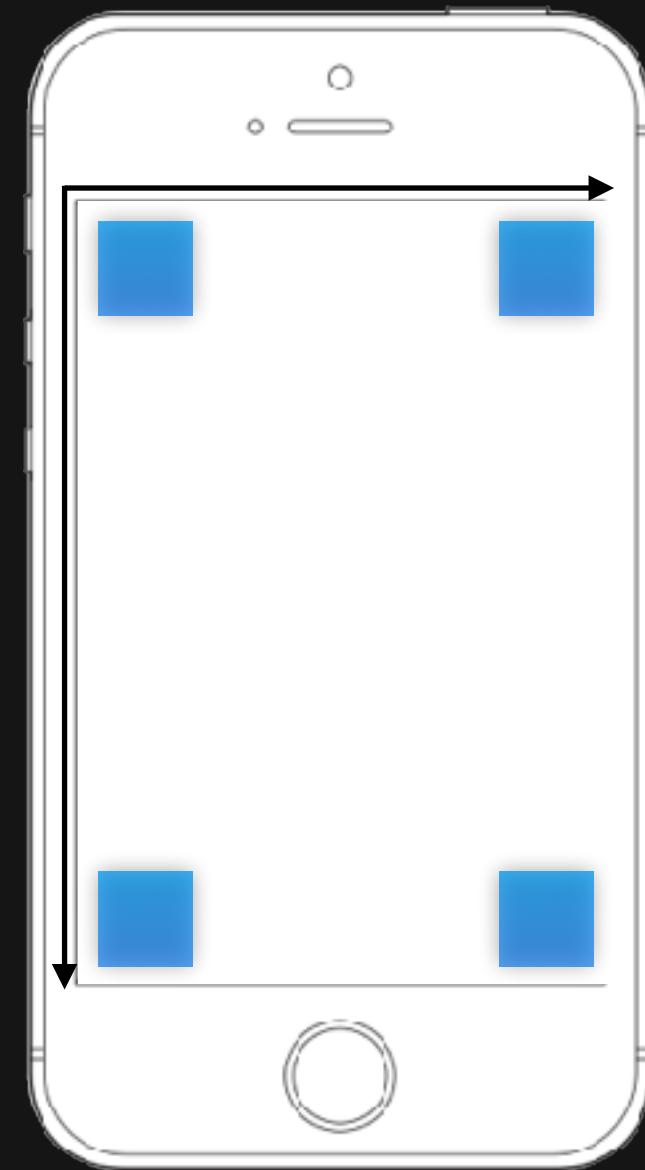


# Auto Layout

- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées

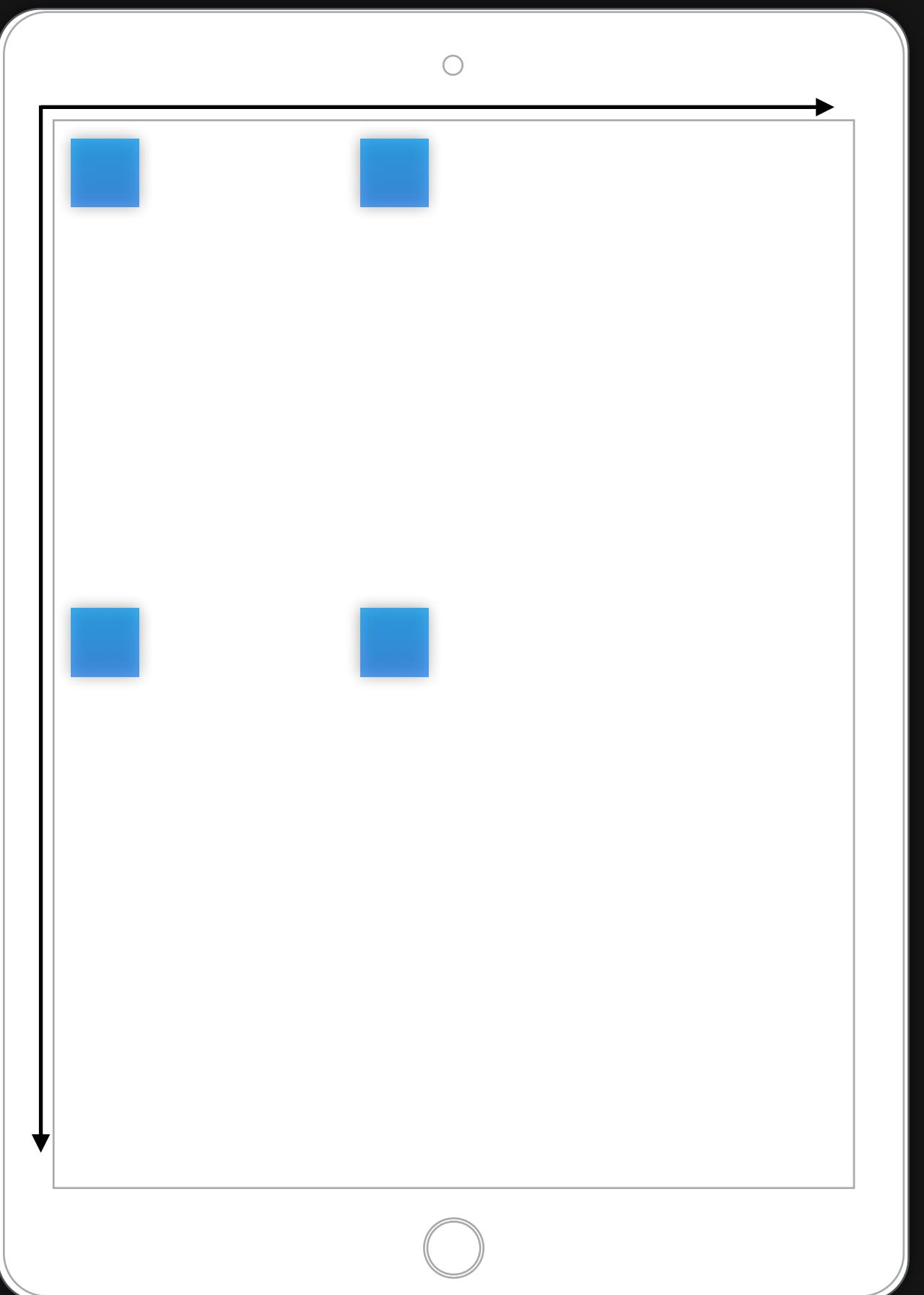
# Auto Layout

- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées



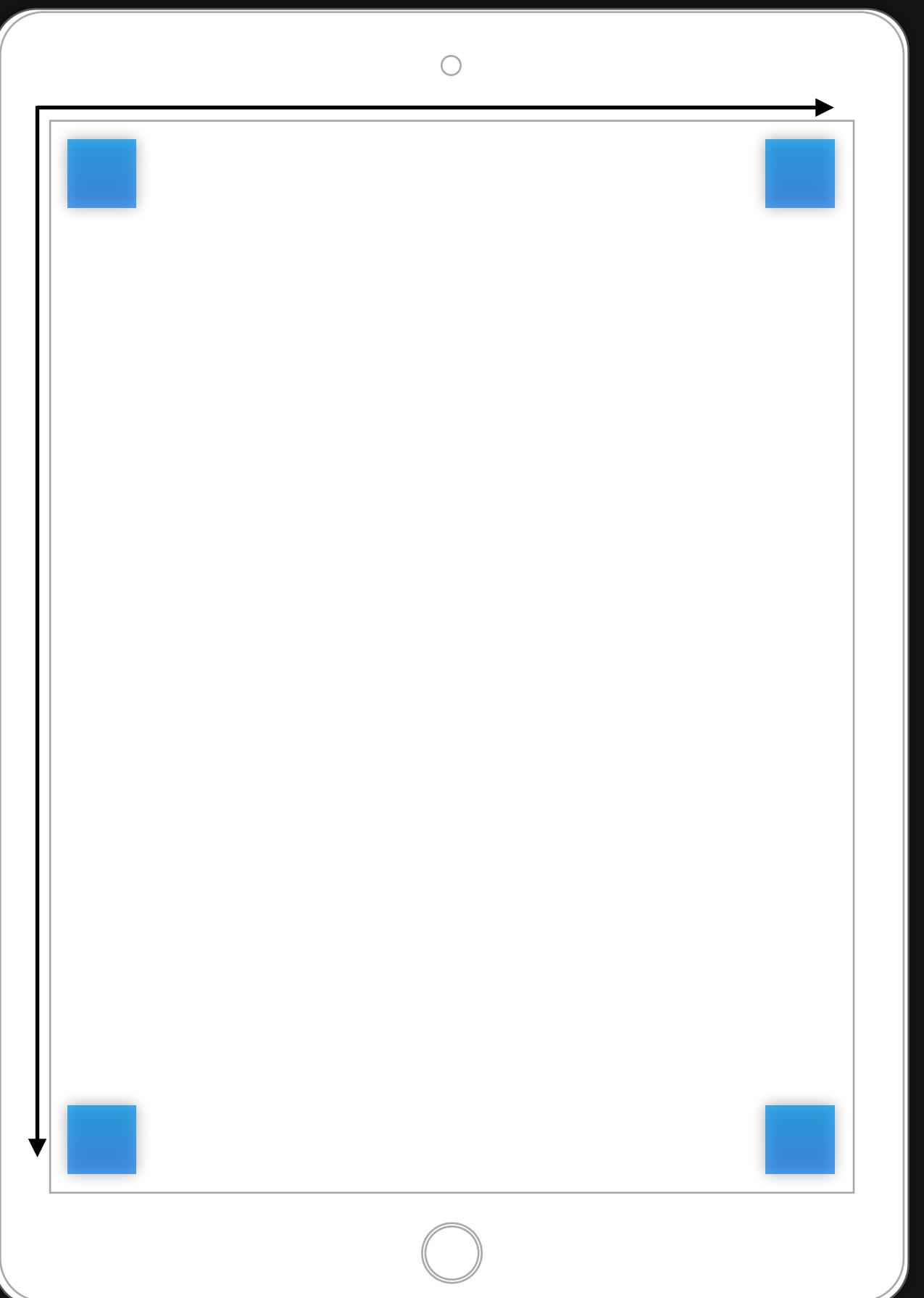
# Auto Layout

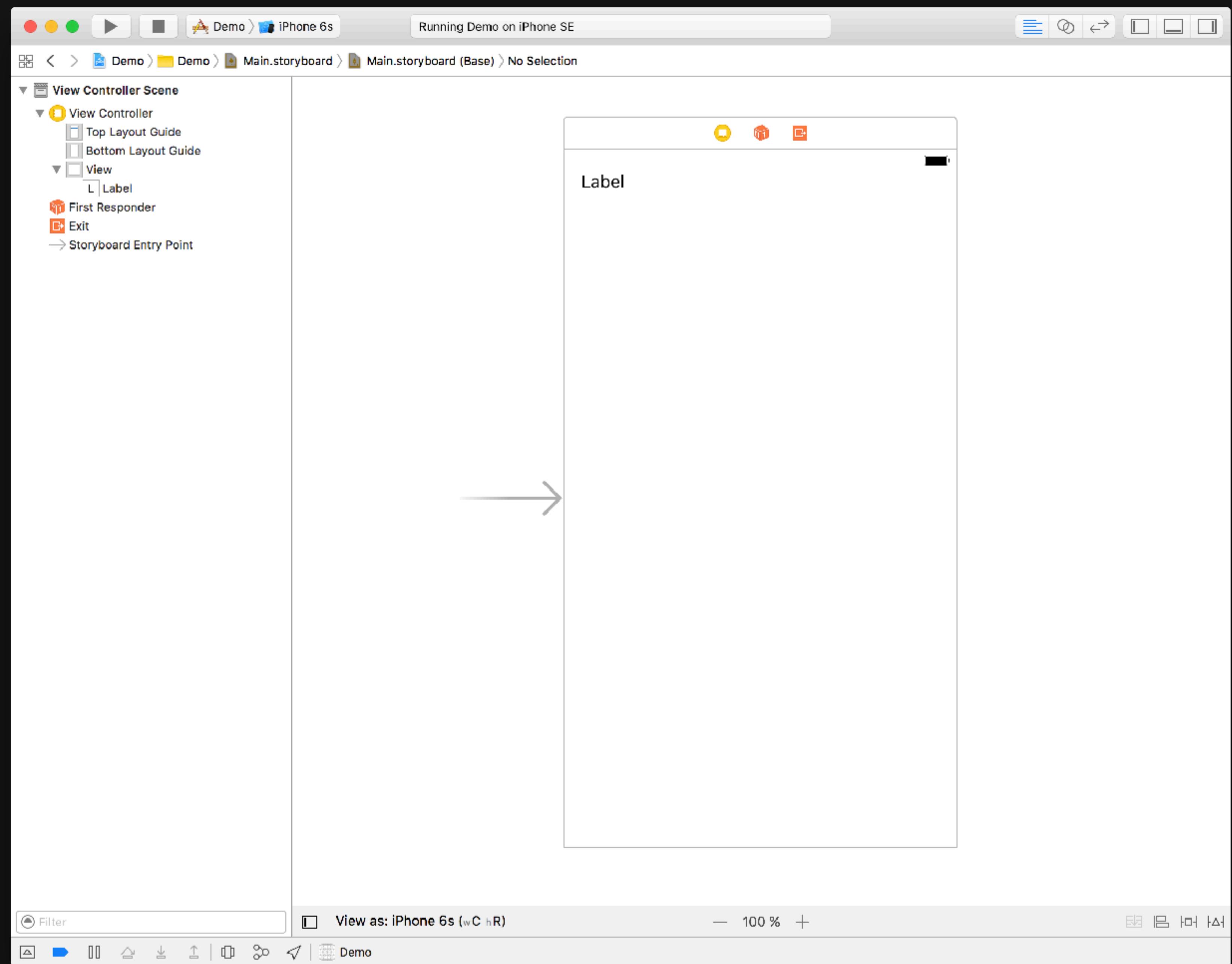
- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées

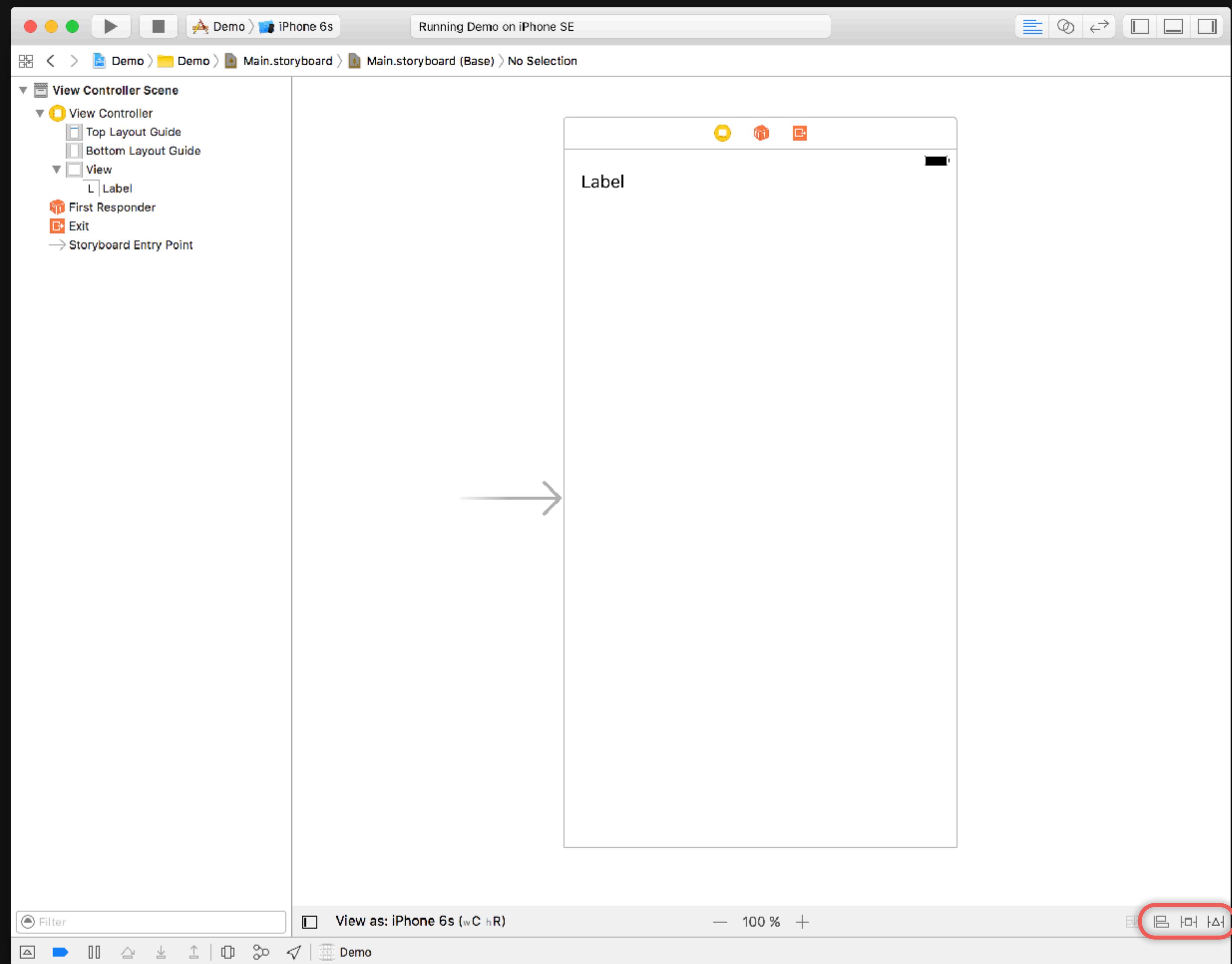


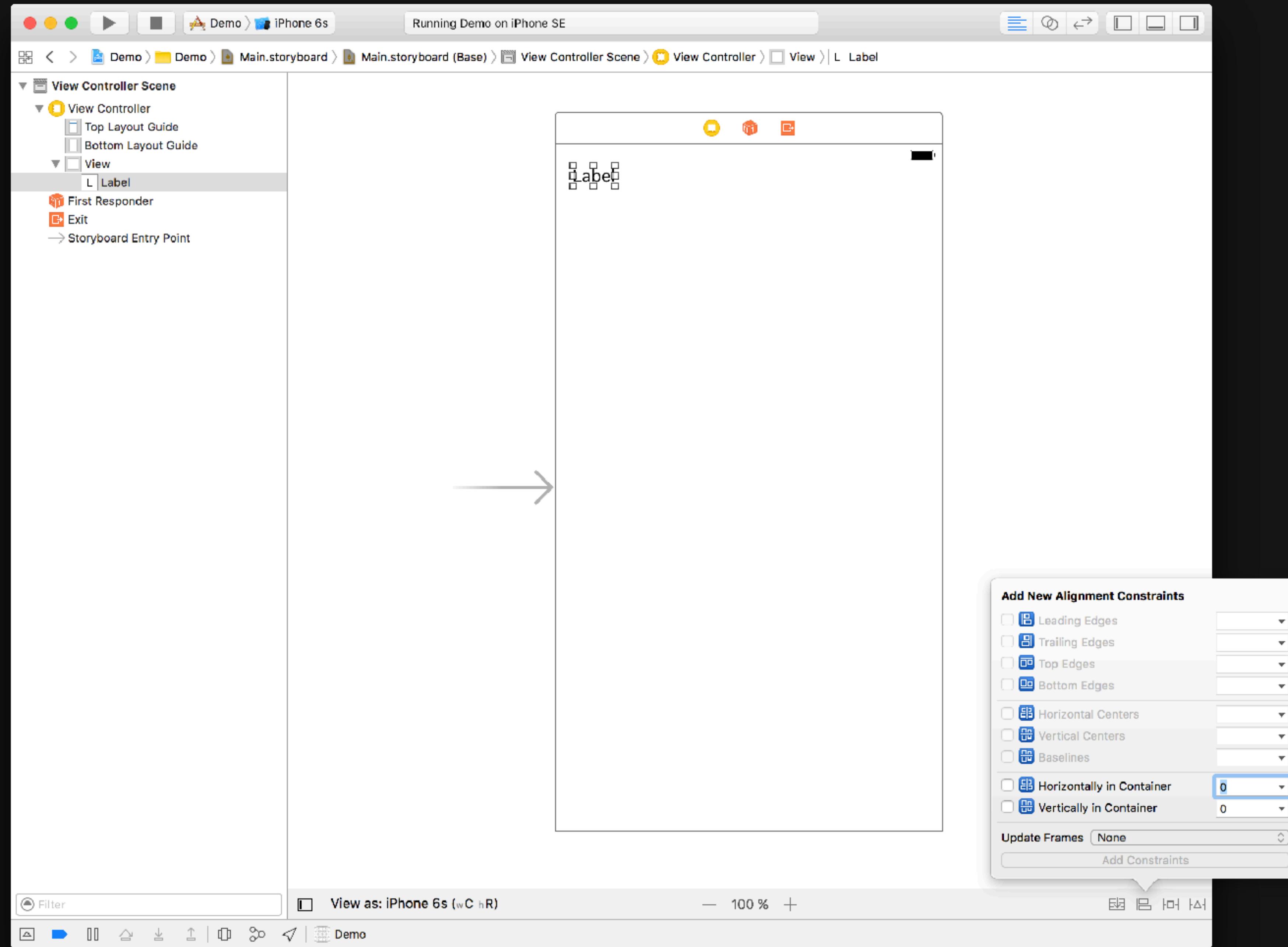
# Auto Layout

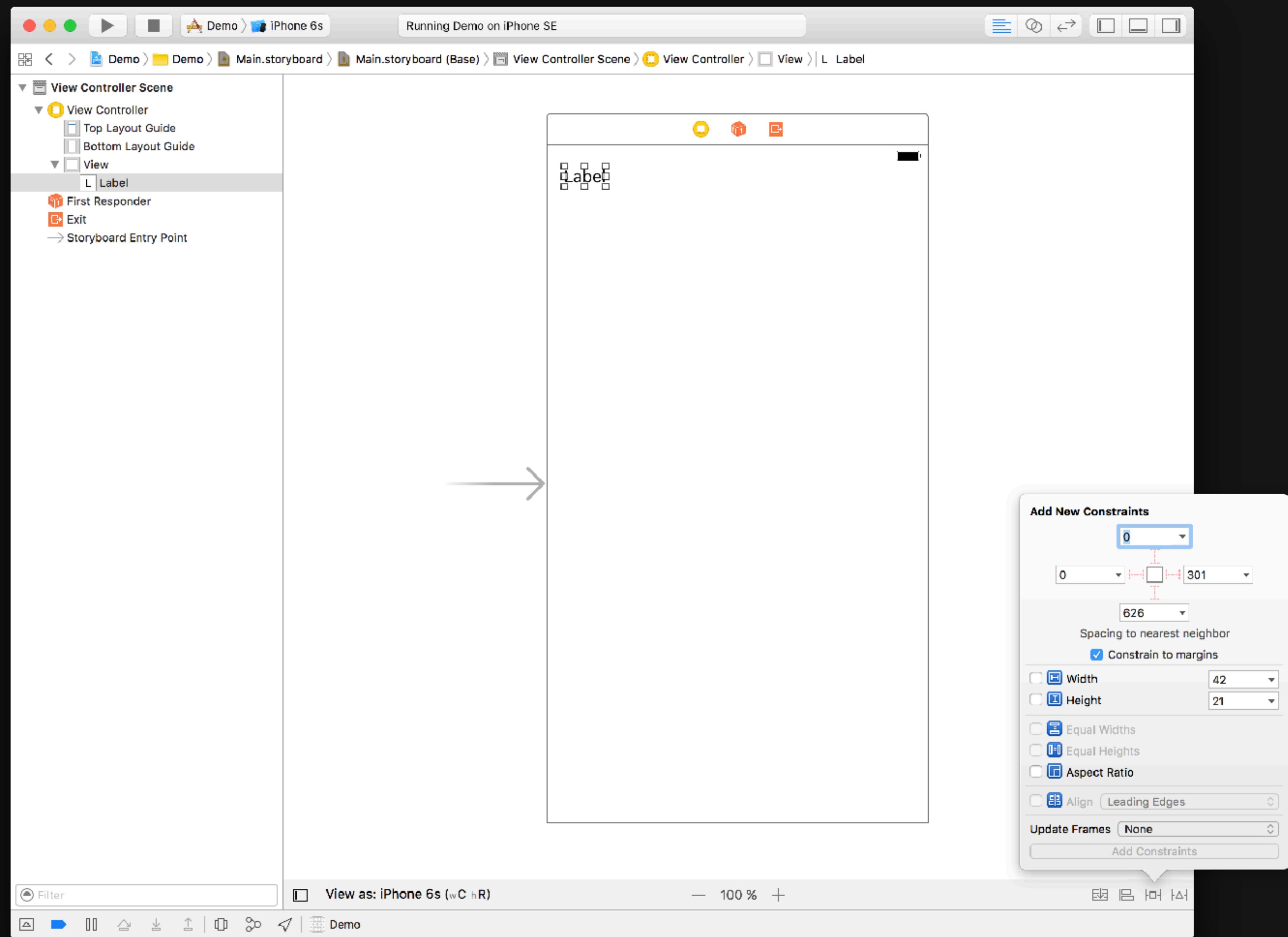
- Par défaut, nos éléments sont positionnés de manière fixe dans le système de coordonnées
- Auto Layout nous permet d'exprimer les positions grâce à des contraintes qui seront évaluées dynamiquement
- Auto Layout définit des relations mathématiques entre nos objets graphiques.
- On doit définir suffisamment de contraintes pour placer les objets sans ambiguïtés

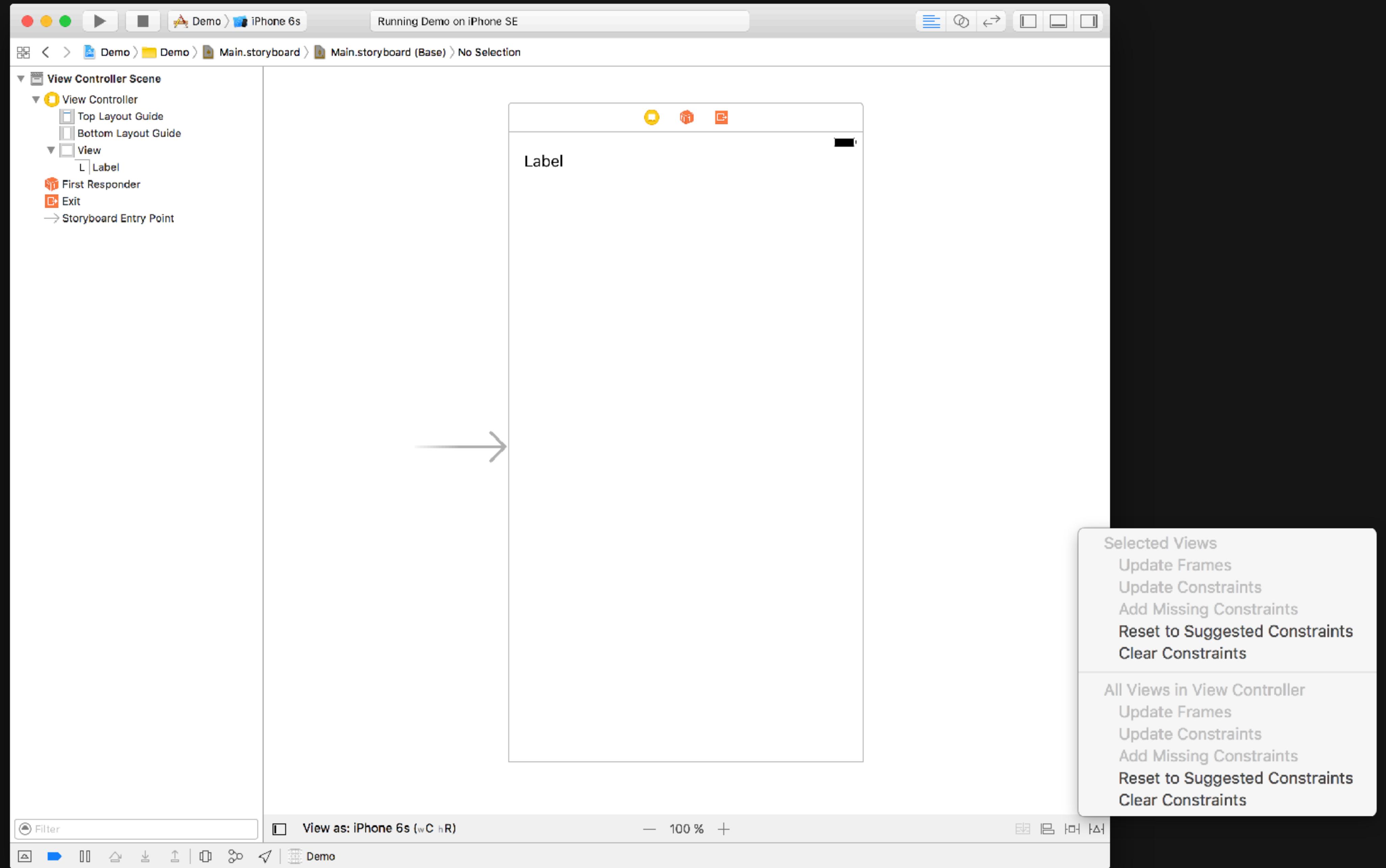












# Auto Layout

- Possibilité de définir les contraintes dans le code
- Contraintes définies sous forme d'une équation linéaire
  - élément1.attribut (=, <=, >=) élément2.attribut x multiplicateur + constante
- NSLayoutConstraint



# Auto Layout

```
convenience init(item view1: Any,  
attribute attr1: NSLayoutAttribute,  
relatedBy relation: NSLayoutRelation,  
toItem view2: Any?,  
attribute attr2: NSLayoutAttribute,  
multiplier multiplier: CGFloat,  
constant c: CGFloat)
```

# Size Classes

- Classification de nos interfaces en fonction de la taille.
- Chaque dimension peut avoir être soit :
  - Compact
  - Regular

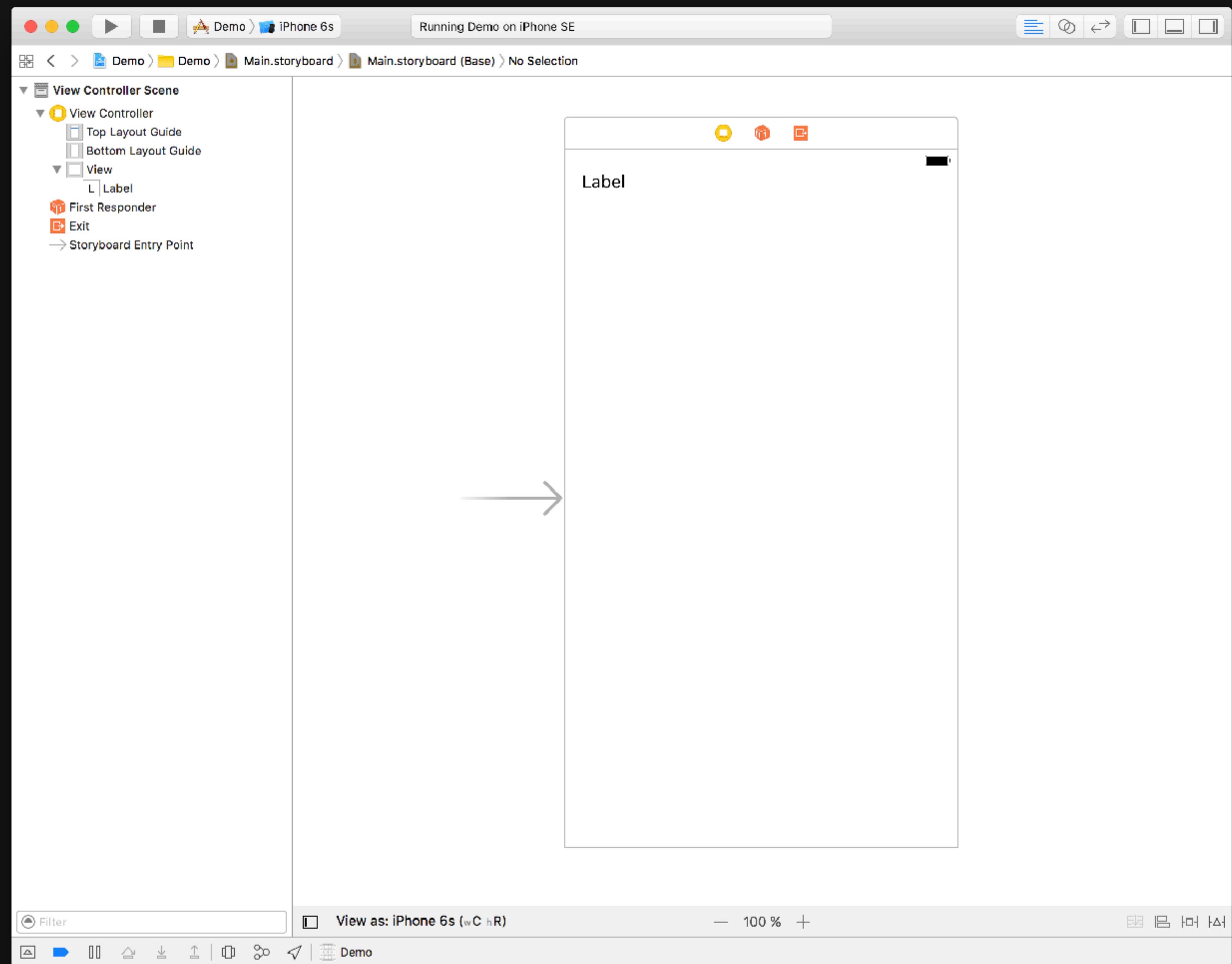
# Size Classes

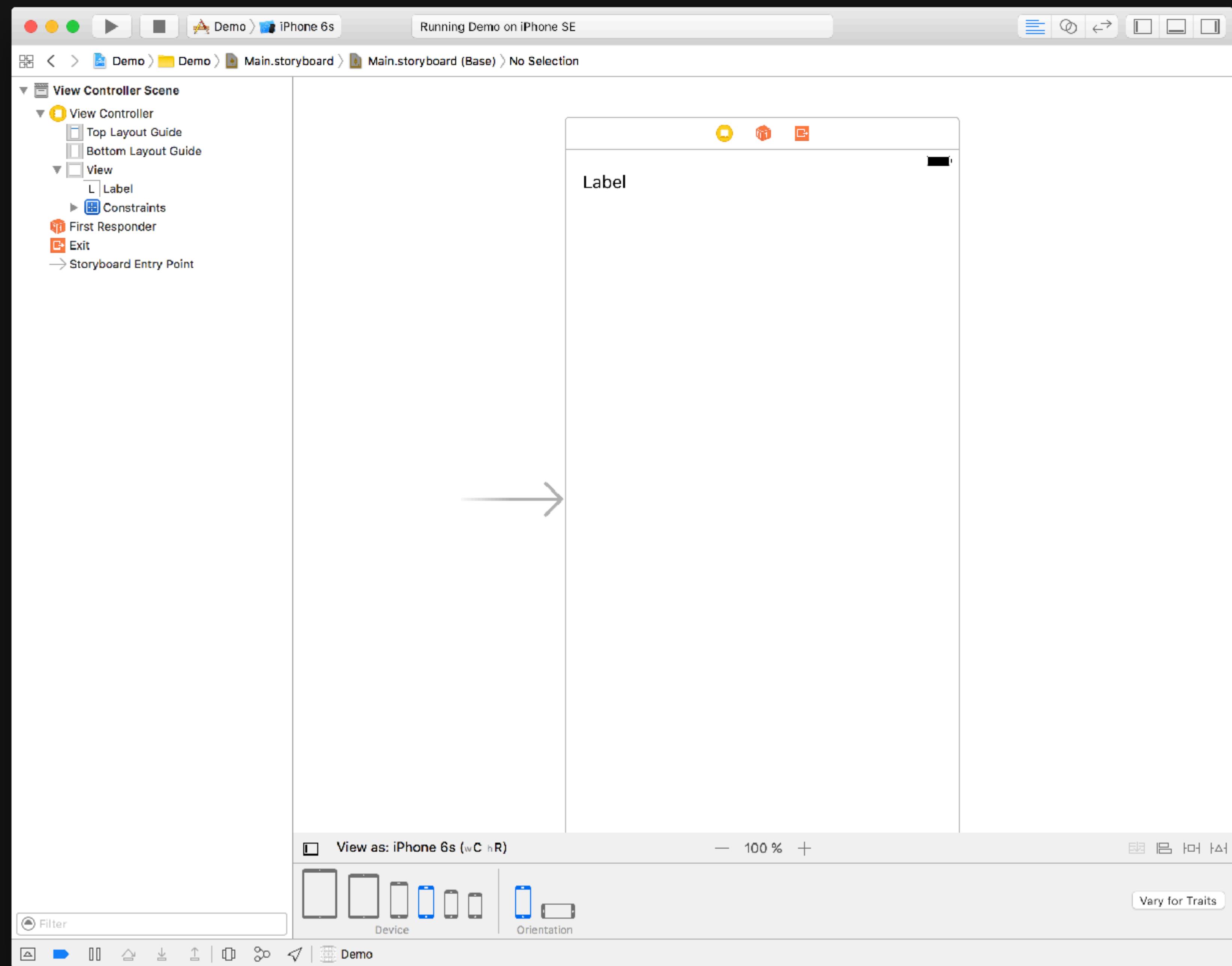
- Depuis le storyboard on peut définir des contraintes différentes pour chaque combinaison de classes
- On peut aussi choisir de retirer ou d'ajouter des composantes graphiques selon les combinaisons de classes

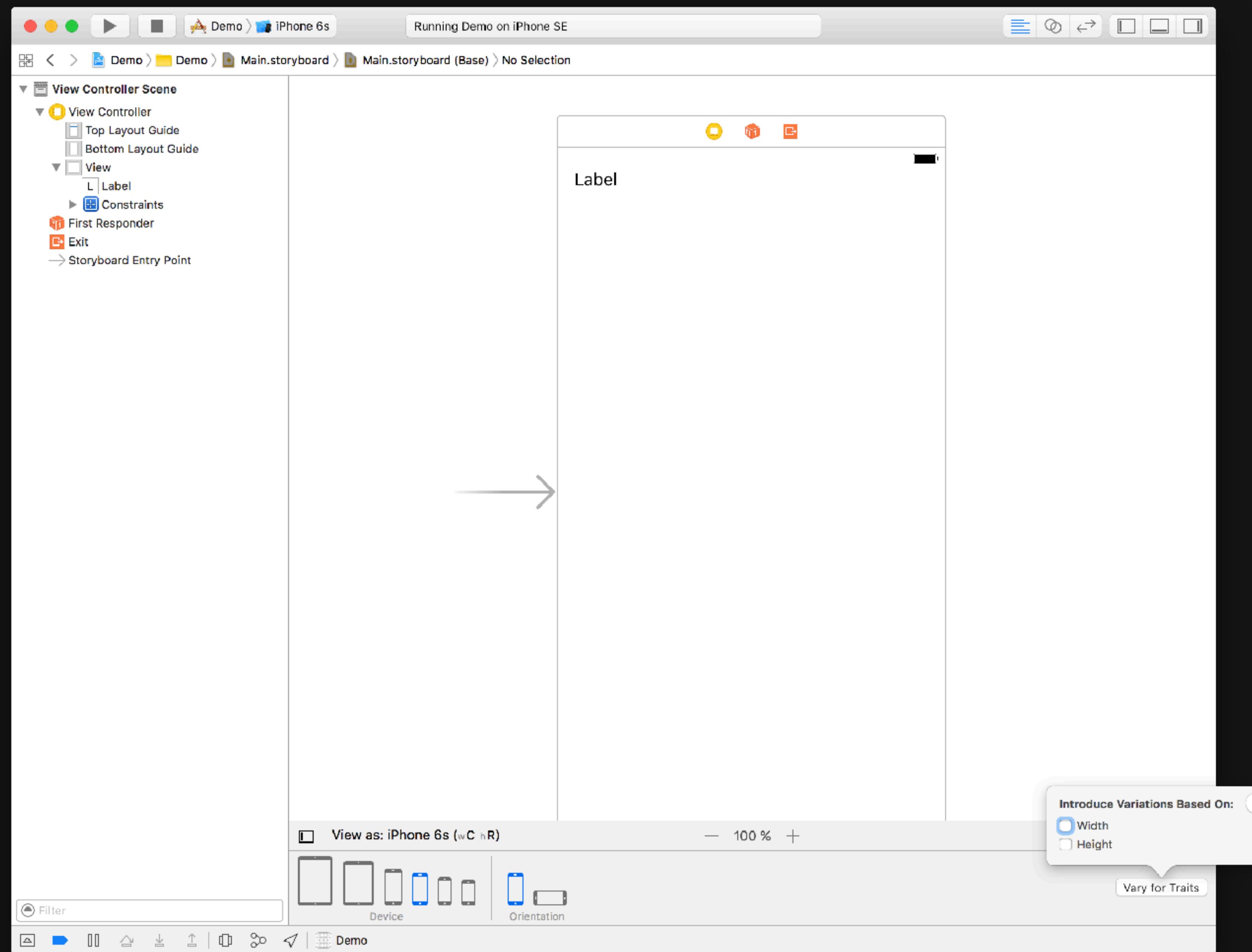
# Size Classes

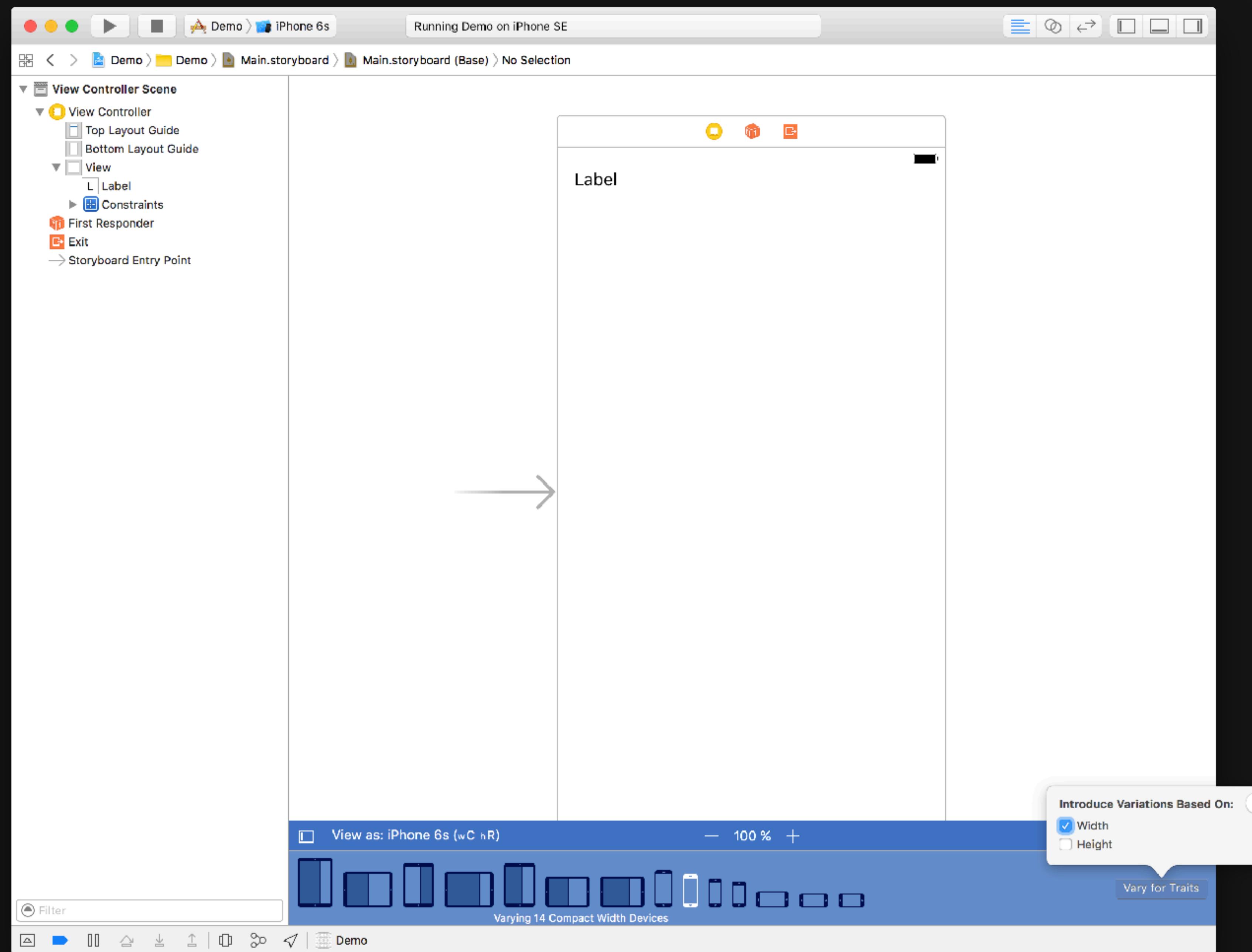
## Size classes horizontales

	Compact	Regular
Compact	iPhone <= 4,7" (Paysage)	iPhone 5,5" (Paysage)
Regular	iPhone (Portrait)	iPad











# Trait collection

- Les UIView, UIViewController, UIPresentationController se conforment à UITraitEnvironment
- Ils exposent une propriété traitCollection

```
var userInterfaceIdiom: UIUserInterfaceIdiom { get }  
var displayScale: CGFloat { get }  
var verticalSizeClass: UIUserInterfaceSizeClass { get }  
var horizontalSizeClass: UIUserInterfaceSizeClass { get }
```

- Peuvent surcharger une méthode pour réagir au changement de classe

```
func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?)
```

# Objets courants



# Boutons

## ▪ UIButton

- `init(type: UIButtonType) -> UIButton`
- `var titleLabel: UILabel? { get }`
- `var imageView: UIImageView? { get }`

Button





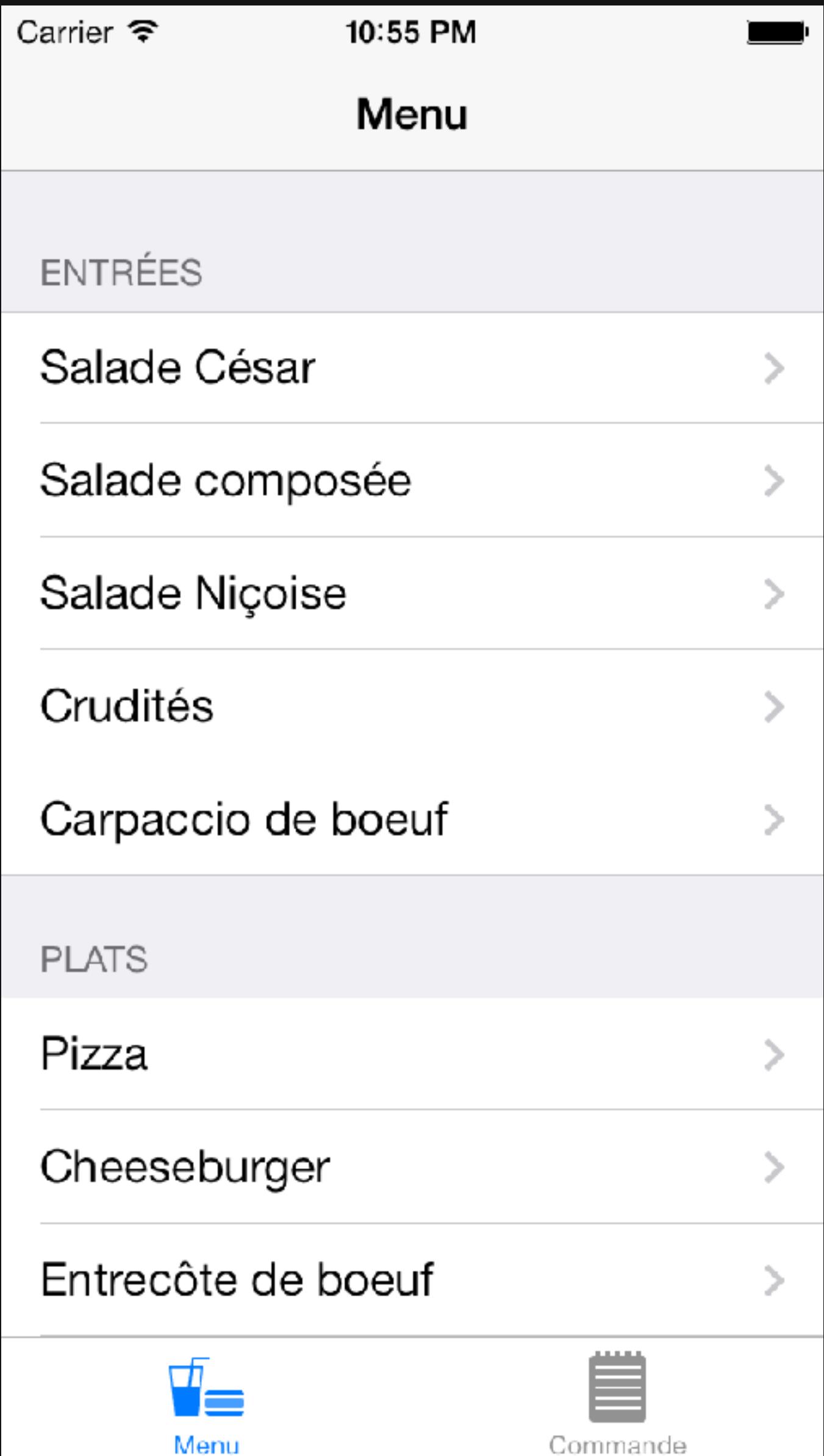
# Boutons

- UIButton → UIButtonTypeCustom
  - `func setImage(_ image: UIImage?, for state: UIControlState)`
- UIControlState
  - .normal
  - .highlighted
  - .disabled
  - .selected

# Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

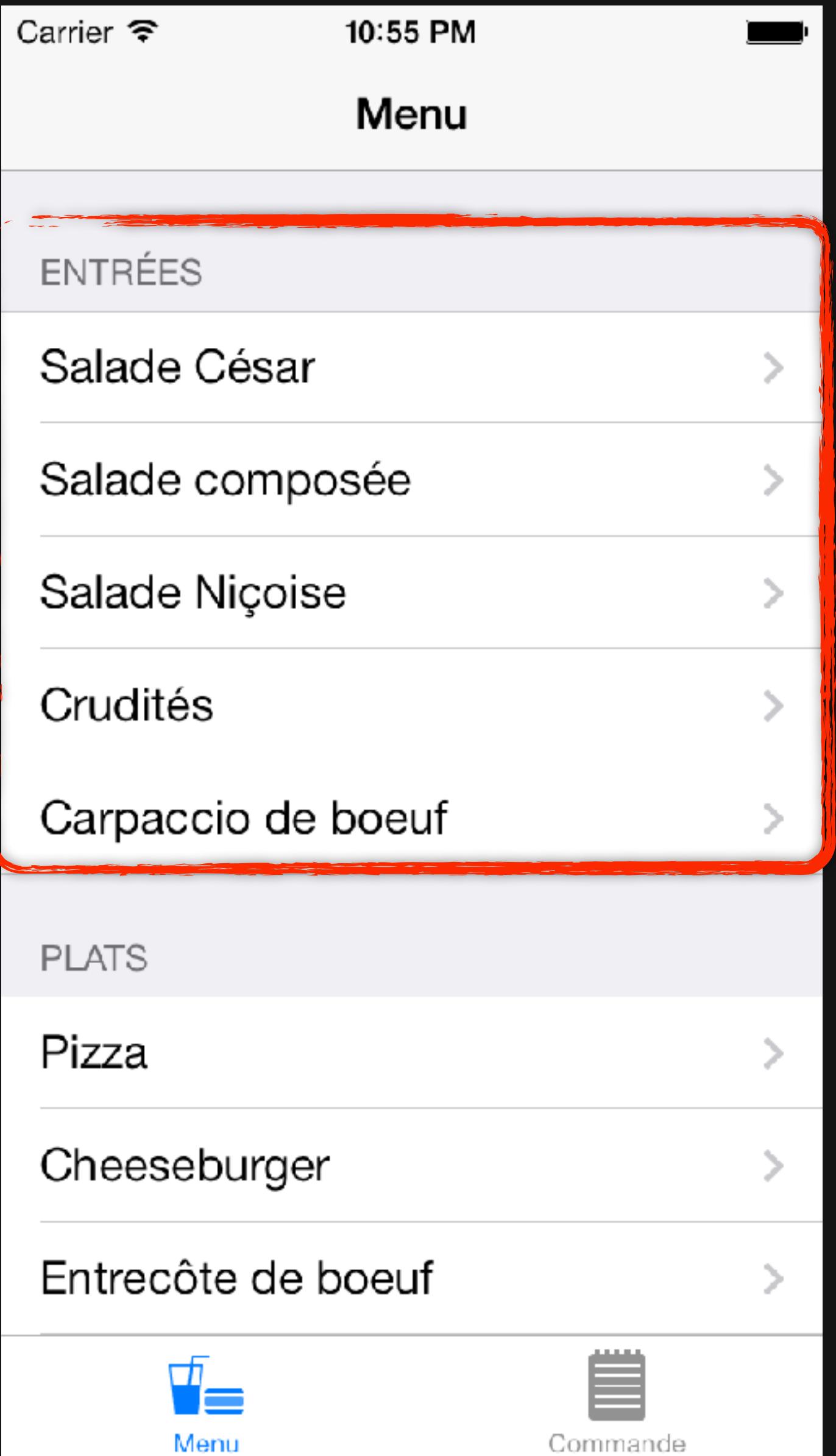


# Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section 0



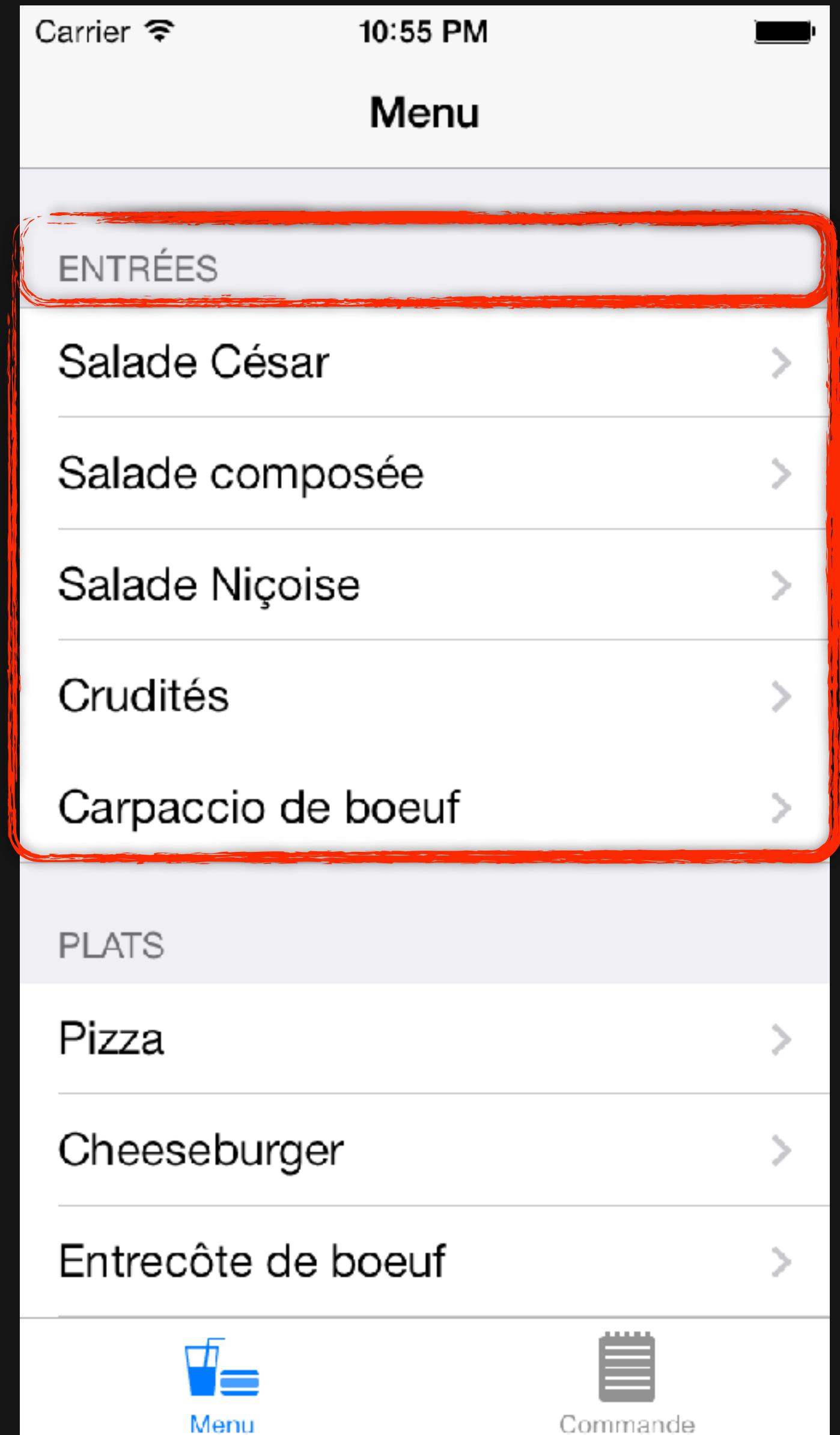
# Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section Header

Section 0



# Table View

- UITableView

- Affiche une liste d'éléments
- Organisation en section
- Deux styles d'affichage
- Contenu statique ou dynamique (via un dataSource)

Section Header

Section 0

Carrier 10:55 PM

Menu

ENTRÉES

Salade César >

Salade composée >

Salade Niçoise >

Crudités >

Carpaccio de boeuf >

PLATS

Cell section 1 line 0

Pizza >

Cheeseburger >

Entrecôte de boeuf >



Menu



Commande

# Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations



La salade porte le nom de Caesar Cardini, un restaurateur italien originaire de la région du Lac Majeur avant d'établir des restaurants à Tijuana et à Ensenada afin de bénéficier d'une clientèle souhaitant contourner la prohibition.

Il existe plusieurs histoires au sujet de la création de cette salade, mais aucune d'elles ne peut être confirmée. La plus commune, racontée par Rosa Cardini (née en 1928), dit que la création fut le résultat d'un épisode de

# Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations



# Navigation Controller

- **UINavigationController**
- Fonctionne avec une pile de ViewControllers
- Gère le titre et le bouton précédent en fonction des propriétés title des ViewControllers
- Gère les animations

Back button



# Tab Bar

- UITabBarController
  - Fonctionne avec un tableau de ViewControllers
  - Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
  - Un TabBar doit toujours rester à l'écran



# Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran



# Tab Bar

- UITabBarController
- Fonctionne avec un tableau de ViewControllers
- Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
- Un TabBar doit toujours rester à l'écran

Tab Bar item  
Tab Bar



# Tab Bar

- UITabBarController
  - Fonctionne avec un tableau de ViewControllers
  - Affiche l'image monochrome de la propriété tabBarItem de chaque ViewController
  - Un TabBar doit toujours rester à l'écran



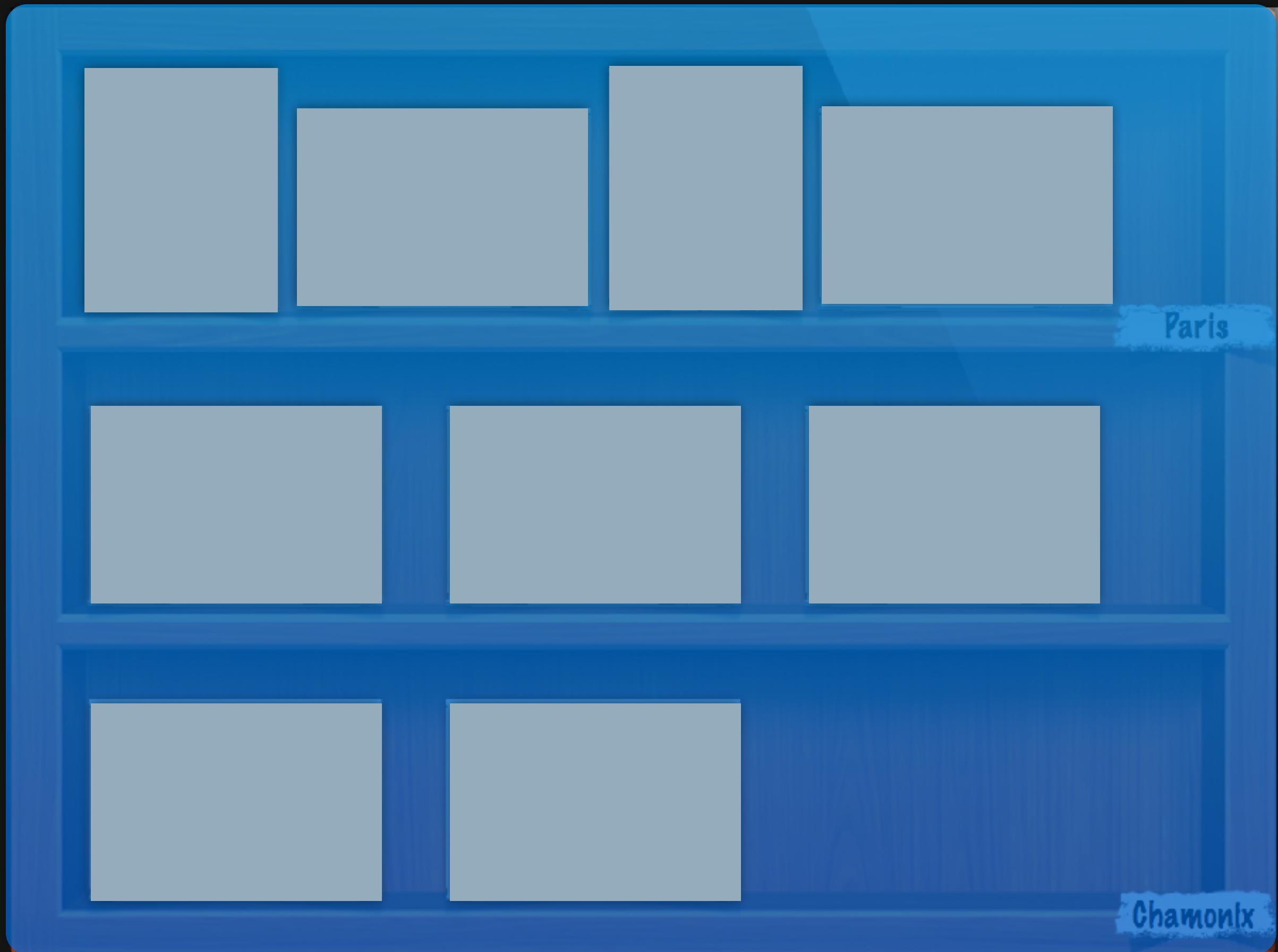
# Collection View

- UICollectionView
- UITableView aux stéroïdes
- Affiche une collection d'éléments
- Complètement personnalisable



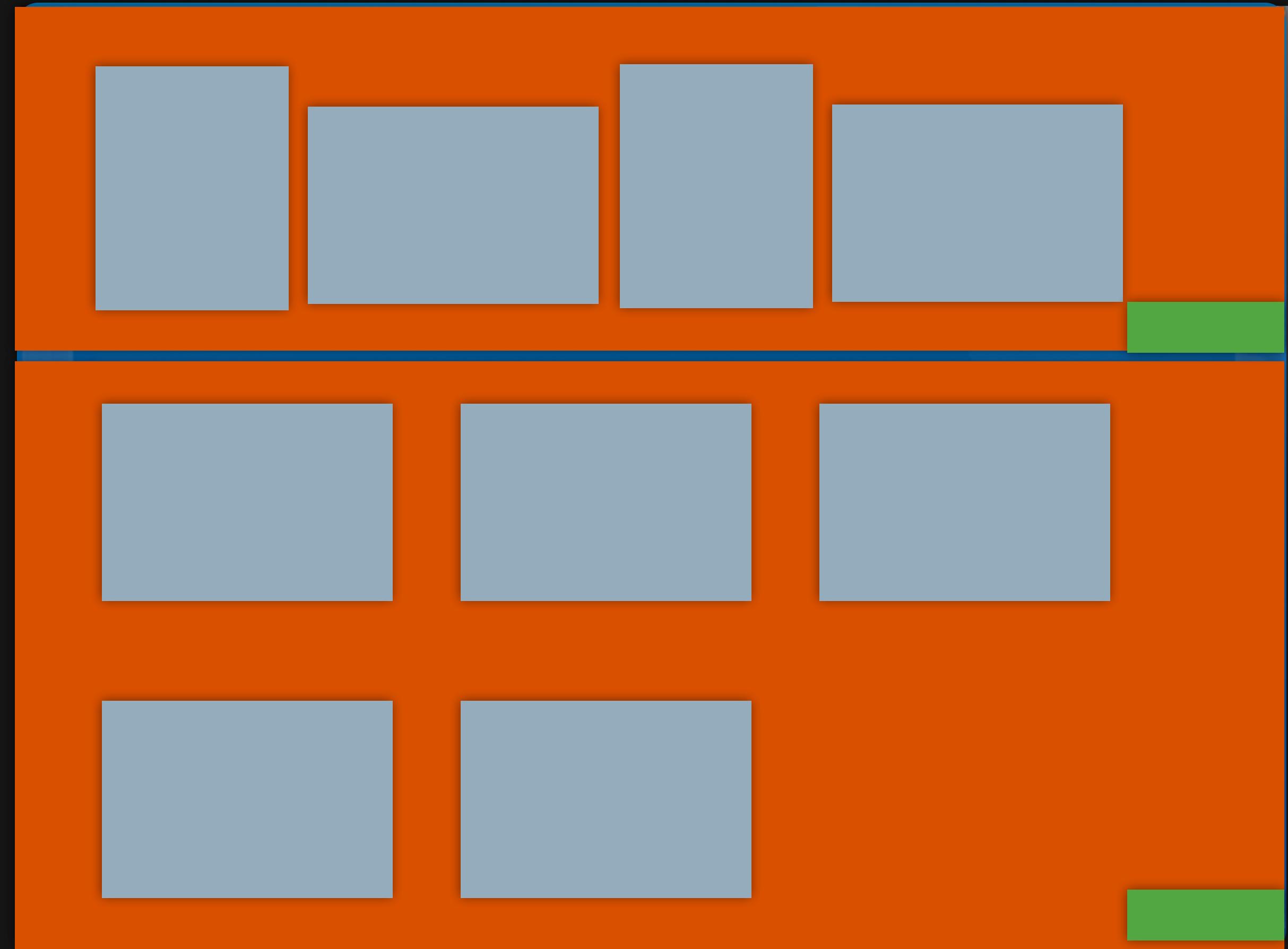
# Collection View

- UICollectionView
  - UITableView aux stéroïdes
  - Affiche une collection d'éléments
  - Complètement personnalisable



# Collection View

- UICollectionView
- UITableView aux stéroïdes
- Affiche une collection d'éléments
- Complètement personnalisable



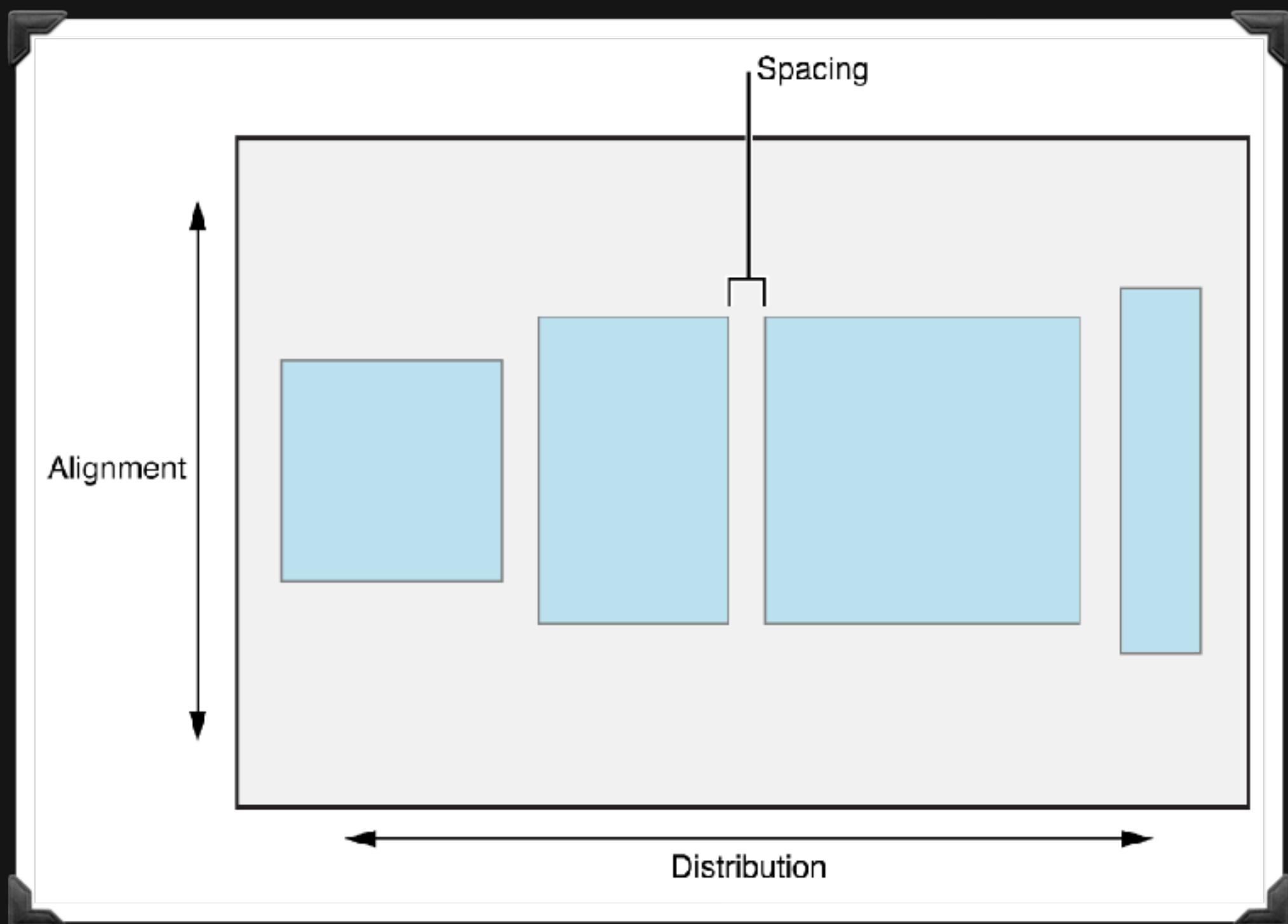
# Collection View

- UICollectionView
- UITableView aux stéroïdes
- Affiche une collection d'éléments
- Complètement personnalisable



# Stack View

- **UIStackView**
  - iOS 9 minimum
  - Gère le layout d'une collection de vues
  - En ligne ou en colonnes
  - Propriétés
    - **axis, distribution, spacing, alignment**



Pour aller plus loin . . .



- Designing for iOS
- UI Elements
- iOS Human Interface Guidelines (iBook)