

Présentation d'iOS

Jean-Ferdy Susini

Maître de Conférences - CNAM

Département Informatique

Sources : Wikipedia, <https://developer.apple.com/>, Xcode

le cnam

Documentation

Paris, 17/11/2017

IOS

2

- Système d’exploitation propriétaire développé par Apple et équipant ses smartphones (iPhones), ses tablettes tactiles (iPads) et ses baladeurs numériques (iPods touch) et plus récemment l’Apple TV et l’AppleWatch
- Sur les bases de Mac OS X (ex NeXT) un développement spécifique orienté par l’IHM
- Premier modèle annoncé en janvier 2007 et commercialisé en juin de la même année.
- <https://developer.apple.com/devcenter/ios/index.action>



Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...

Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...



Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...



Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...



Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...
→ écran capacitif multi-points
surface reconfigurable en entrée et en sortie



Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...
 - écran capacitif multi-points
surface reconfigurable en entrée et en sortie
 - contrainte mémoire (souvent limitée au regard des applications desktop)

Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...
 - écran capacitif multi-points
surface reconfigurable en entrée et en sortie
 - contrainte mémoire (souvent limitée au regard des applications desktop)
 - contrainte CPU (architecture différente)

Repenser le smartphone

3

- Développer pour un système mobile :
 - La taille de l'objet -> taille de l'écran et des touches ou autres systèmes de saisie...
 - ➡ écran capacitif multi-points
surface reconfigurable en
entrée et en sortie
 - contrainte mémoire (souvent limitée au regard des applications desktop)
 - contrainte CPU (architecture différente)
 - ➡ architecture matérielle haut de gamme

Repenser le smartphone

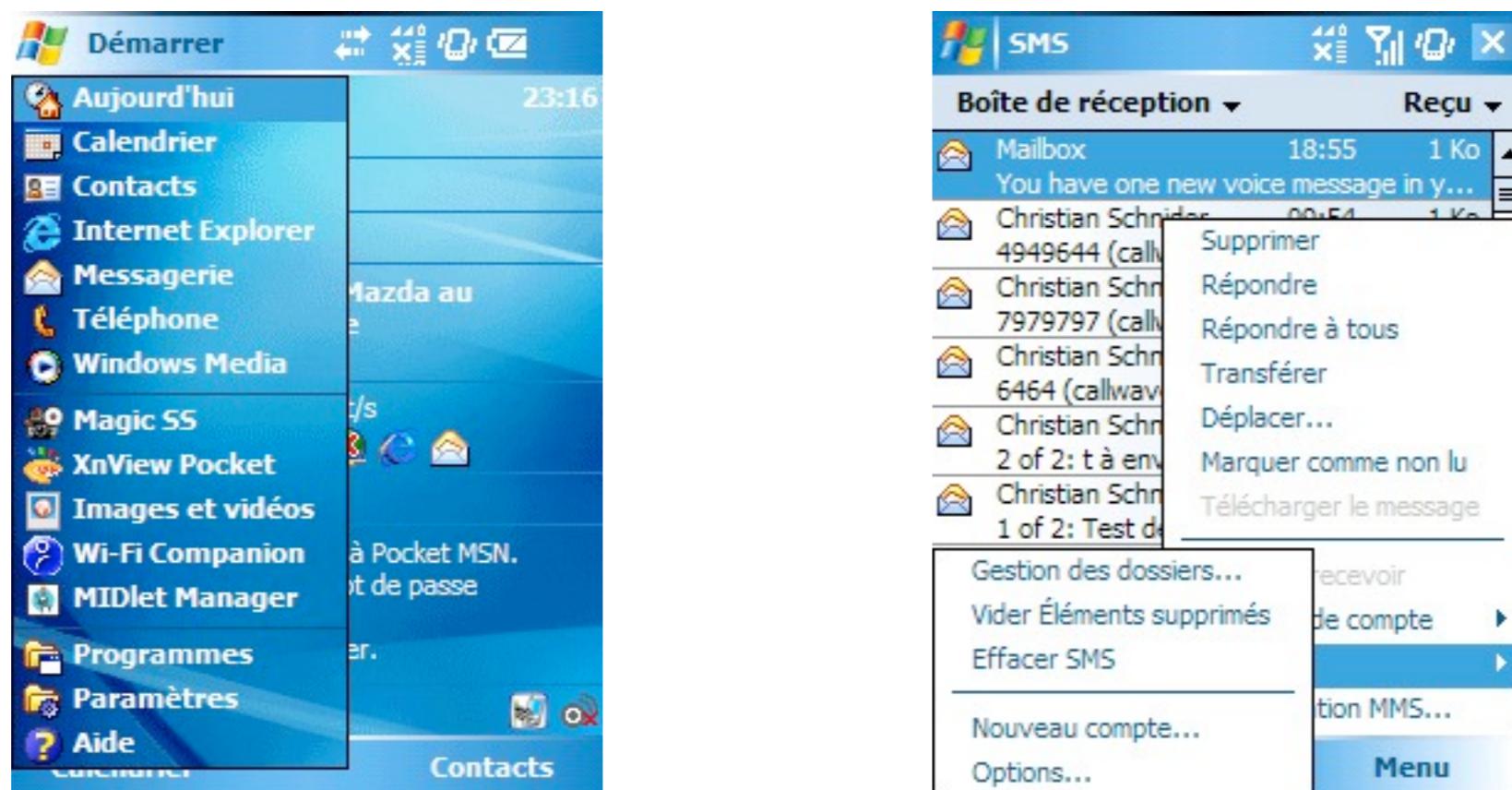
4

- La métaphore du bureau passe mal à l'échelle du smartphone

Repenser le smartphone

4

- La métaphore du bureau passe mal à l'échelle du smartphone



Repenser le smartphone

4

- La métaphore du bureau passe mal à l'échelle du smartphone
 - Une application à la fois

Repenser le smartphone

4

- La métaphore du bureau passe mal à l'échelle du smartphone
 - Une application à la fois
 - Un seul écran à la fois

Repenser le smartphone

4

- La métaphore du bureau passe mal à l'échelle du smartphone
 - Une application à la fois
 - Un seul écran à la fois
 - Une seule tâche à la fois

Repenser le smartphone

4

- La métaphore du bureau passe mal à l'échelle du smartphone
 - Une application à la fois
 - Un seul écran à la fois
 - Une seule tâche à la fois
- Nouvelle interface tactile

Repenser le smartphone

→ Nouvelle interface tactile

Repenser le smartphone

5

→ Nouvelle interface tactile

Repenser le smartphone

5

→ Nouvelle interface tactile



Repenser le smartphone

Repenser le smartphone

6

→ Nouvelle interface tactile

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback
- Tenir compte du contexte d'utilisation

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback
- Tenir compte du contexte d'utilisation
 - être informé du contexte : capteurs...

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback
- Tenir compte du contexte d'utilisation
 - être informé du contexte : capteurs...
 - s'adapter au contexte : affichage, saisie de données automatisée...

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback
- Tenir compte du contexte d'utilisation
 - être informé du contexte : capteurs...
 - s'adapter au contexte : affichage, saisie de données automatisée...
 - importance des animations

Repenser le smartphone

6

→ Nouvelle interface tactile

- Faire la part belle au contenu
- Peu d'éléments d'interaction à l'écran
- Peu d'aide => cohérent et prédictible, feedback
- Tenir compte du contexte d'utilisation
 - être informé du contexte : capteurs...
 - s'adapter au contexte : affichage, saisie de données automatisée...
 - importance des animations

→ Métaphore mécanique

L'évolution

7



- Juin 2007, iPhone (E ou 2G) : 480x320 (163ppi)
iOS 1.0 -> 3.1.3, EDGE, accéléromètres,
triangulation GSM/WiFi, luminosité/proximité,
2MPx
- sept. 2007, iPod Touch : iOS 1.1.3 -> 3.1.3
- juillet 2008, iPhone 3G : 480x320 (163ppi) iOS
2.0 -> 4.2.1, 3G, GPS
- sept 2009, iPod Touch 2 : IOS 2.1.1 -> 4.2.1
- juin 2009, iPhone 3GS : 480x320 (163ppi) iOS
3.0 -> 6.1.6, HSDPA, BT 2.1, boussole, 3.2Mpx,
autofocus...
- sept 2009, iPod Touch 3 : iOS 3.1.1 -> 5.1.1

L'évolution

8



- juin 2010, iPhone 4 : 960x640 (326ppi, Retina)
iOS 4.0 -> 7.1, APN 2,5Mpx, flash-LED,
gyroscope, camera frontale, 2 micros
- sept 2010, iPod Touch 4 : iOS 4.1 -> 6.1.6
- Oct. 2011, iPhone 4s : 960x640 (326ppi) iOS 5.0
-> 9.3.5, proc. Dual-core, APN 8Mpx,
GLONASS, coprocesseur audio, Siri...
- Sept. 2012, iPhone 5 : 1136 x 640 (326ppi) iOS
6.0 -> 10.3.3, 4G LTE, FC 1.2Mpx
- oct 2012 : iPod Touch 5 : iOS 6.0 -> 9.3.5
- Sept. 2013, 5s : 1136 x 640 (326ppi) iOS 7.0 ->
10.3.3, capteur TouchID, μ P 64 bits, M7
coProcesseur capteurs, True Tone, burst-mode...
5c : basé sur l'ancien iPhone 5.



L'évolution

9

- sept 2014 : l'iPhone 6 : 4,7 pouces, 1334×750 (326 ppi), l'iPhone 6+ : 5,5 pouces, 1920x1080 (401ppi), iOS 8, 11.1, baromètre, coprocesseur dédiés aux capteurs (M8), NFC, stabilisation optique APN arrière.
- avril 2015, AppleWatch : 272×340 (326 ppi), 312 × 390 (326 ppi), watchOS 1 -> 4, proc. S1, 512 Mo RAM, 8Go, cardio fréquence, centrale inertielle, WiFi, Bluetooth, NFC, force touch.



L'évolution



- sept 2015 : l'iPhone 6s : 4,7 pouces, 1334×750 (326 ppi), 6s+ : écran 5,5 pouces, 1920x1080 (401ppi), iOS 9 -> 11.1, proc. A9, 2Go de mémoire vive, APN arrière 12Mpx, APN avant 5Mpx. Coprocesseur M9, 3Dtouch avec retour optique.
- sept 2016, iPhone 7 : 4,7 pouces, 1334×750 (326 ppi), iPhone 7+ : écran 5,5 pouces, 1920x1080 (401ppi) iOS 10-> 11.1, proc. A10 (4 cœurs 64 bits), 3Go de mémoire vive, étanchéité, FC 7Mpx.
- sept 2016, AppleWatch 2 : watchOS 3.2.2 -> 4, proc. S1 ou S2 dual core, GPS, mise en place progressive de l'ApplePay...

L'évolution

11

- 
- sept 2017, iPhone 8 : 4,7 pouces, 1334×750 (326 ppi), iPhone 8+ : écran 5,5 pouces, 1920x1080 (401ppi), iOS 11 -> 11.1, proc. A11 (6 cœurs 64 bits), Neural Engine, 2Go de mémoire vive, Galileo, QZSS.
 - sept 2017, AppleWatch 3 : watchOS 4, proc. S3 dual core, réseau cellulaire (e-SIM)...
 - sept 2017, iPhone X : 5,8 pouces 2436×1125 (458ppi) iOS 11 -> 11.1, proc. A1, FaceID, Animoji
- 

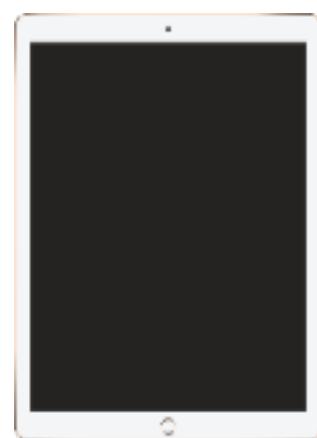
iPad

12

- avril 2010, iPad : 1024x768 (132 ppi) iOS 4.2.1 -> 5.1.1 (accel, boussole, luminosité, WiFi, BT, 3G/AGPS)
- mars 2011, iPad 2 : iOS 4.2.1 -> 9.3.5 (Dual-core, cameras, gyroscope)
- mars. 2012, iPad Retina : 2048x1536 (264 ppi) iOS 5.1.1 -> 9.3.5 (BT 4, 3GLTE, GLONASS)
- oct. 2012, iPad 4 : 2048x1536 (264 ppi) IOS 6.0 -> 10.3.3
- oct. 2012 iPad Mini : 1024x768 (163 ppi) IOS 6.0 -> 7.0
- nov. 2013 iPad Air : 2048x1536 (264 ppi) iOS 7-> 11.1, A7 - 64 bits
- nov. 2013 iPad Mini (2nd G) : 2048x1536 (326ppi) IOS 7->11.1, A7-M7...



iPad



- oct. 2014 iPad Air 2 : 2048x1536 (264 ppi) iOS 8.1 -> 11.1, baromètre, APN arrière 8Mpx, touchID.
- oct. 2014 iPad Mini (3rd G) : 2048×1536 (326ppi) iOS 8 -> 11.1, A7-M7, touchID
- oct. 2015 iPad Mini (4rth G) : 2048×1536 (326ppi) iOS 9->11.1, A7-M7, APN arrière 8Mpx, touchID
- Nov. 2015 iPad Pro : 12,9 pouces 2 732×2 048 (264ppi) 9,7 pouces 2048×1536 (264ppi), iOS 9.1 -> 10.1, touchID, 4 hauts parleurs, ApplePencil et Clavier
- Juin 2017, iPadPro : 12,9 pouces 2 732×2 048 (264ppi) 10,5 pouces 2048×1536 (264ppi) iOS 10.3.2 -> 11.1, proc. A10X 6 coeurs.



Matériel et Logiciel évoluent ensembles



- sept 2010, ATV : IOS 4.1 -> ATV 6.2.1 (iOS 7.1.2)
- mar. 2012, ATV : IOS 5.1 -> ATV 7.2.2 (iOS 8.4.2)
- jan. 2013, ATV rev. A: ATV 5.2 -> ATV 7.2 (iOS 8.3)
- oct. 2015, ATV : ATV 9 -> ATV 11 (iOS11)
- sept. 2017, ATV (4k) : ATV 11 (iOS11)
- Hardware et Software développés de concert
- iTunes et les mises à jour OTA depuis la version 5 facilitent les transitions
- Très faible fragmentation : iOS 11 sorti en sept. 2017 représente déjà 58%, avec iOS 10 : 27%.

11.X	58,1 %
10.X	27,2 %
9.X	12,2 %
8.X ...	0,5 %

Matériel et Logiciel évoluent ensembles

15

- 12 résolutions et 6 ratios, mode portrait ou paysage

ratio 4/3	Classic	Retina
iPhone-iPod	480x320	960x640
iPhone-iPod 5, C, CE...	1136x640	
iPad-iPad mini	1024x768	2048x1536
iPad Pro	-	2732x2048 ou 2224x1668

autres ratios	Retina
iPhone 6,6s,7, 8	1334x750
iPhone 6+, 6s+,7+, 8+	1920x1080
Apple Watch	272x340 ou 312x390
iPhoneX	2436x1125

- Emulation des résolutions et ratios sur iPad, iPhone 5, 5s, 5c, CE puis sur iPhone 6, 6s, 6+, 6s+, 7, 7+, 8, 8+...
- Peu de versions différentes de l'OS (1 seul constructeur et 1 seul éditeur et peu de possibilités de personnalisation). Support de matériel assez ancien.

Architecture du système

Core OS

Architecture du système

16



BaseBand (gestion radio GSM, WiFi, BT)
Darwin OS : noyau XNU (mach 3.0+BSD)
Free BSD POSIX, IOKit, Launchd...

Core OS

Architecture du système

16

Nombreux services de bas niveaux
bibliothèques C
Objective-C runtime

Core Services

Core OS

Architecture du système

16

Services d'accès aux données de types
audio/vidéo...
(écrit en C ou en Objective-C)

Media Framework

Core Services

Core OS

Architecture du système

Ensemble de Frameworks Objective-C
structurant la programmation des applications

16

Cocoa Touch

Media Framework

Core Services

Core OS

Architecture du système

16

Cocoa Touch

Media Framework

Core Services

Core OS

Les Frameworks

17

Cocoa Touch

Les Frameworks

17

Cocoa Touch

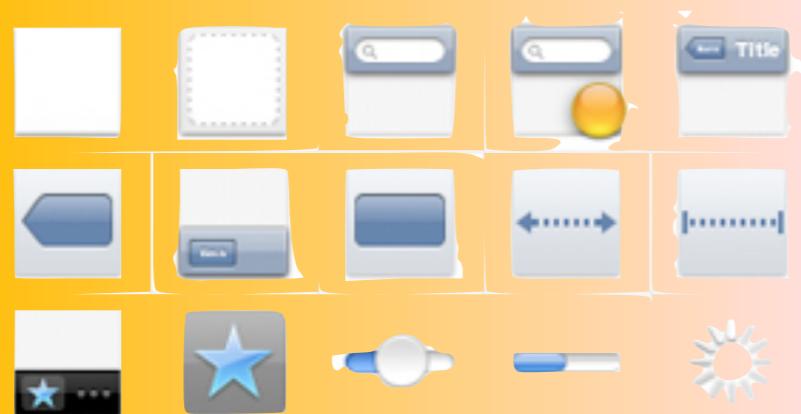
Les Frameworks

17

Cocoa Touch

UIKit

Interface utilisateur
Gestion des
applications, des
événements...



Les Frameworks

17

Cocoa Touch

UIKit

Interface utilisateur
Gestion des
applications, des
événements...



Foundation

Classes utilitaires
gestion des collections
Objets interfaces avec
les services sous-jacents

Les Frameworks

18



Address Book UI, Event Kit UI, Game Kit, iAd, Map Kit, Message UI, Twiter, UIKit, WebKit



Asset Library, AV Foundation, Core Audio, Core Graphics, Core Image, Core Midi, Core Text, Core Video, Image I/O, GLKit, MediaPlayer, Open AL, OpenGL ES, Quartz Core



Account, AddressBook, Ad Support, CFNetwork, Core Data, Core Foundation, Core Location, Core Media, Core Motion, Core Telephony, Event Kit, Foundation, Mobile Core Services, PassKit, Newsstand Kit, Quick Look, Social, Store Kit, System Configuration



Accelerate, Core Bluetooth, External Accessory, Generic Security Services, Security,



Le langage Objective-C

19

- Le développement sous iOS repose comme sous MacOS principalement sur l'utilisation d'un langage spécifique : le langage Objective-C
- Crée au début des années 80, il s'agit d'un langage Orienté Objet (à base de classes) dans la lignée de SmallTalk dont il reprend des éléments de syntaxe et les principaux concepts comme l'héritage simple, les protocoles (interfaces)...
- Construit au-dessus de C dont il est un sur-ensemble strict => est utilisé dans le cœur même de l'OS. Les drivers sont également développés grâce à des frameworks Objective-C (IOKit...).

Le langage Objective-C

20

- Un runtime dynamique intégré dès le début (approche très différente de C++, son concurrent de l'époque), faiblement typé (**id**), par la suite ont été rajoutées des annotations de type.
- Pas de GC, mais gestion de la mémoire par compteurs de références (manuelle puis automatique ARC). Produit du code natif, éventuellement plusieurs codes pour supporter différentes architectures processeur. (mach-o)
- Objective-C distingue deux phases dans la communication entre les objets :
 - le *messaging* : envoi d'un message à un objet
 - la sélection de la méthode : un type particulier existe pour gérer la sélection (SEL : ~ pointeur de fonction)

Objective-C

21

- Envoie d'un message à un objet reprend la syntaxe de Smalltalk, si l'objet ne sait pas quoi faire du message, il ne fait rien :

[receiver message]

- utilisation d'arguments

[receiver message:arg]

[receiver message:arg1 argument2: arg2]

[receiver message: arg1, arg2, nil]

- valeur de retour d'un message

int res = [[receiver message] message2];

Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...

Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...



Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
 - Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...

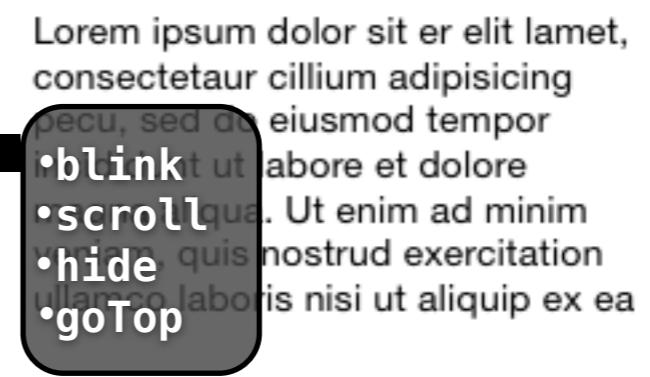


Text
Lorem ipsum dolor sit er elit lamet, consectetur cillum adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...



Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...



Lorem ipsum dolor sit er elit lamet,
consectetaur cillum adipisicing
pecu, sed do eiusmod tempor
ut labore et dolore
• **blink**
• scroll
• hide
• goTop

Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...



- SEL : blink
- target: o1

Lorem ipsum dolor sit er elit lamet,
consectetaur cillum adipisicing
pecu, sed do eiusmod tempor
incididunt ut labore et dolore
magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea

Objective-C

22

- Héritage simple et pourtant arbre de classes peu profond
=> L'héritage n'est pas le mécanisme le plus utilisé :
 - protocol et délégation : permet d'offrir une réponse proche de l'héritage multiple
 - catégories : spécialisation de code existant même si on a pas accès au code source de ces objets
- Popularise quelques cadres (patrons de conception) de programmation : MVC, Target/Action, delegation...



- SEL : blink
- target: o1

Lorem ipsum dolor sit er elit lamet,
consectetaur cillum adipisicing
pecu, sed do eiusmod tempor
incididunt ut labore et dolore
magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea

Construction d'IHM

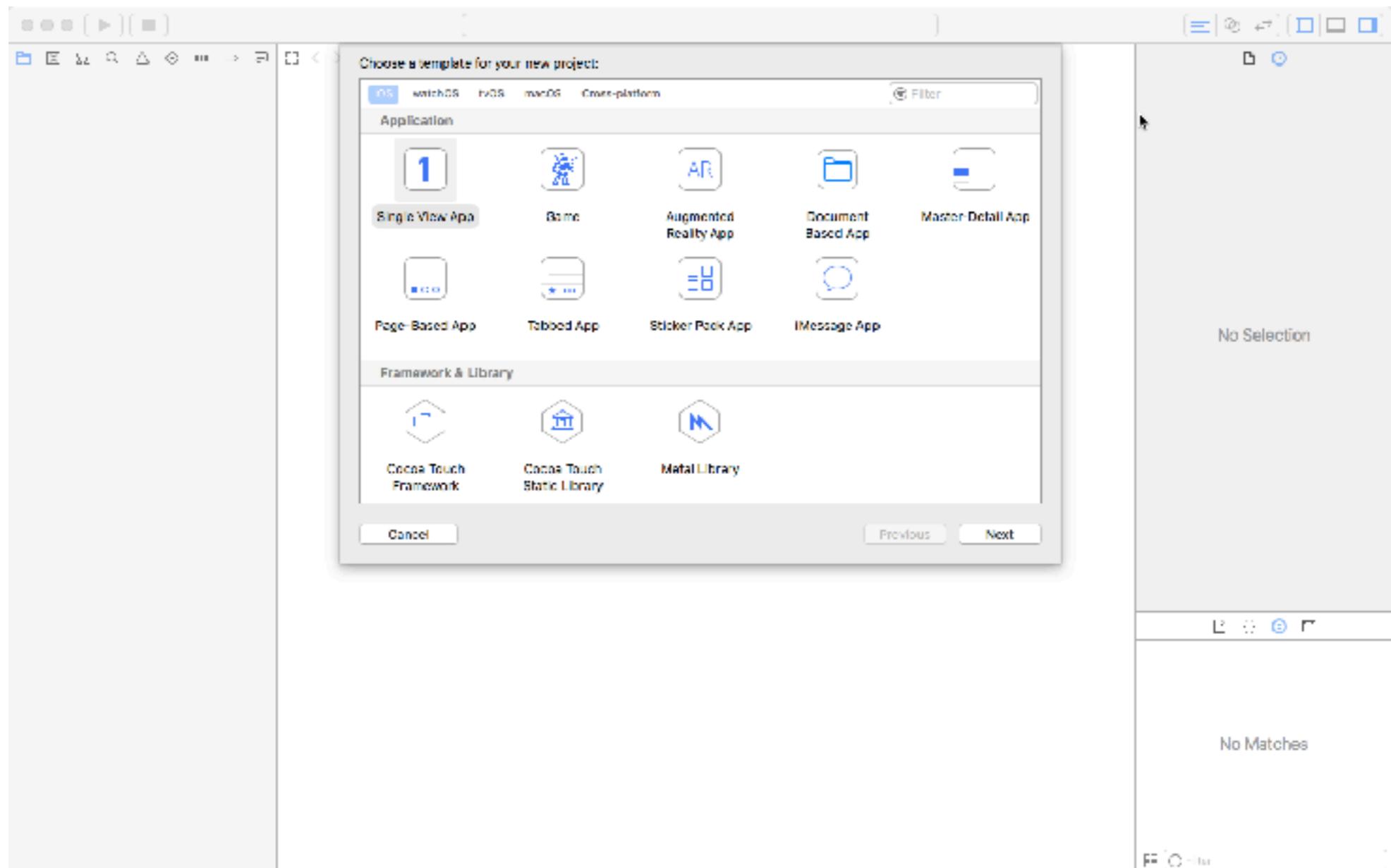
23



- description basée sur des fichiers ressources binaires (.nib) ou XML (.storyboard)
- construction programmatique également possible
→ iOS Human Interface Guidelines

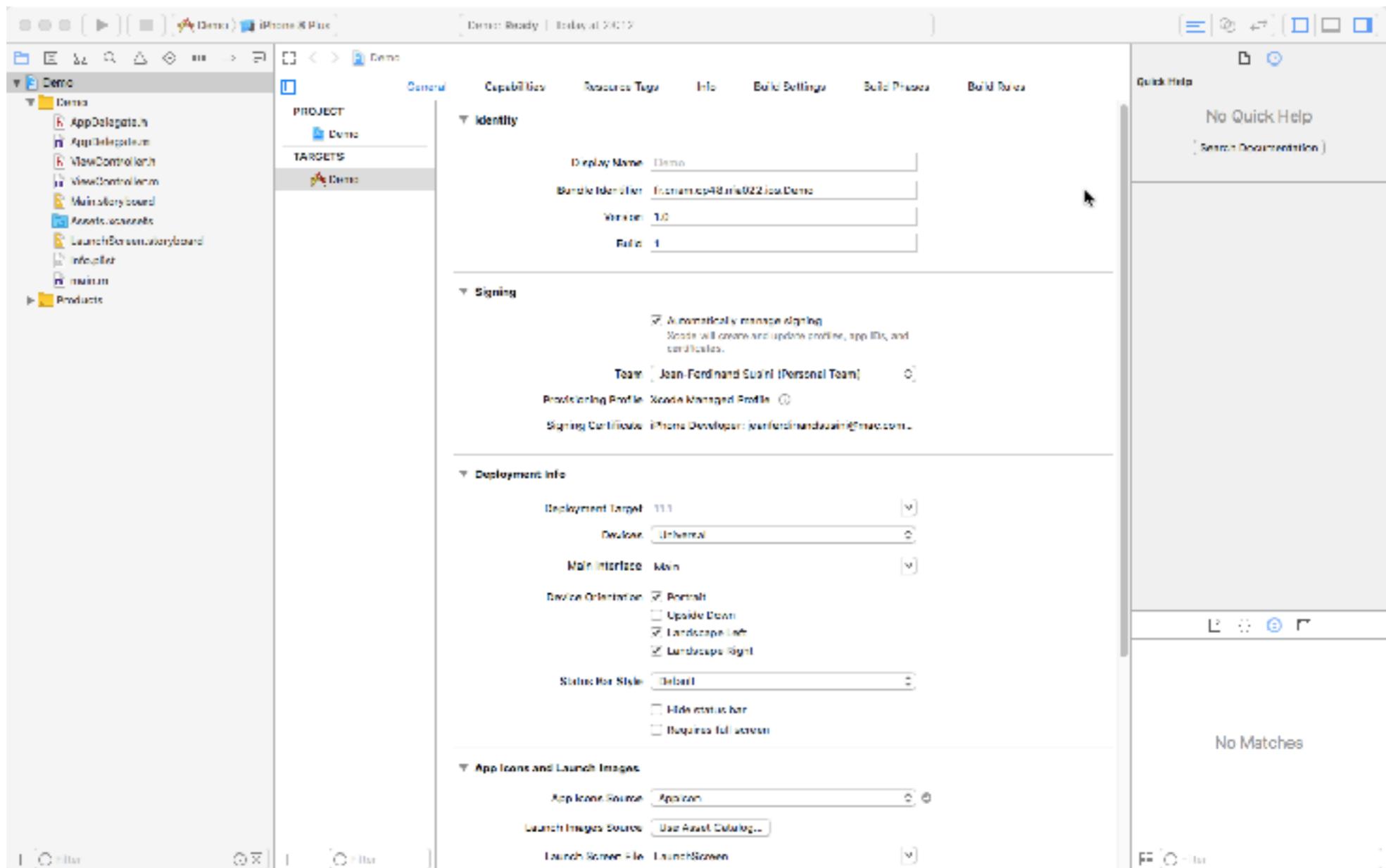
Démo Xcode 9

24



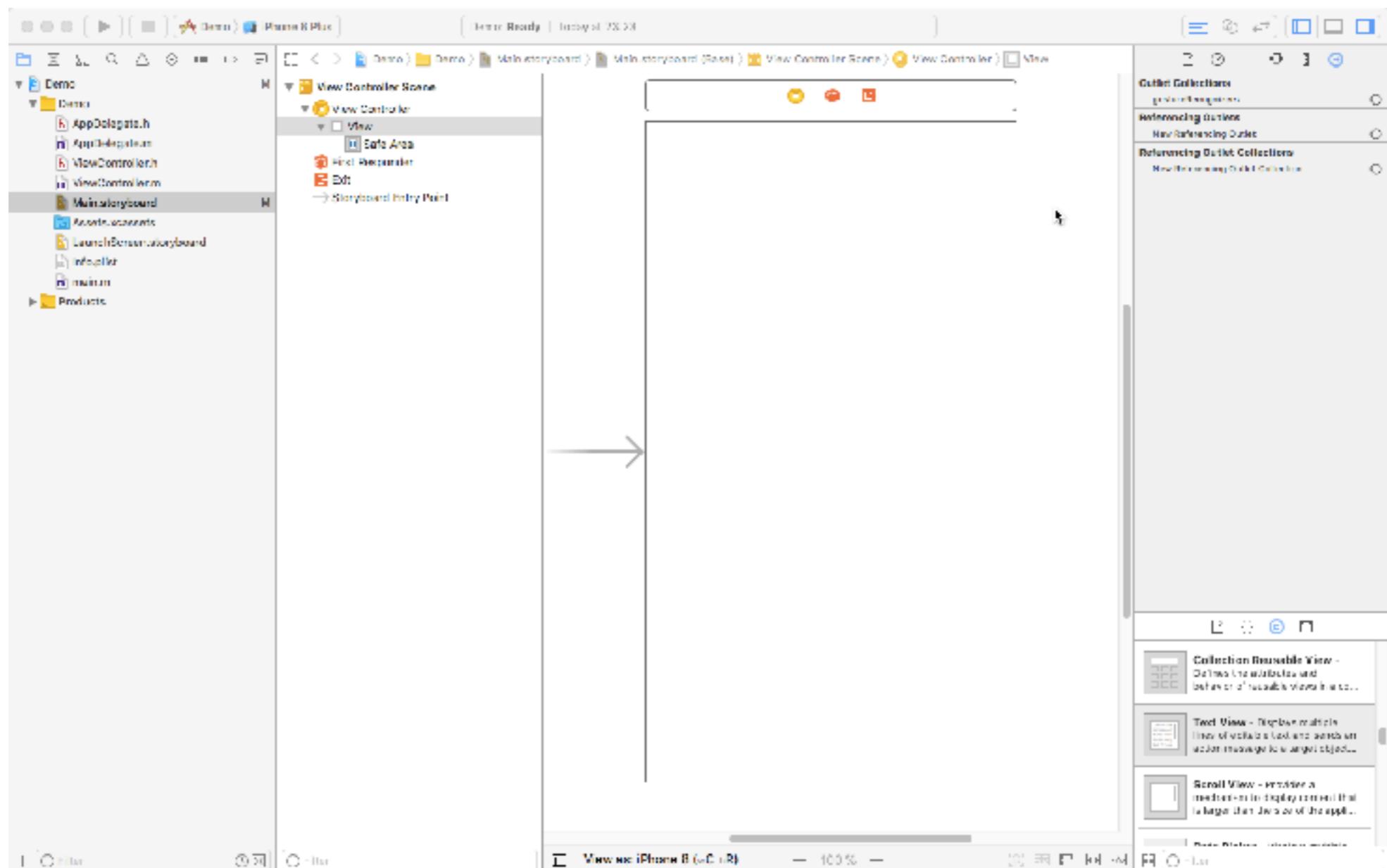
Démo Xcode 9

24



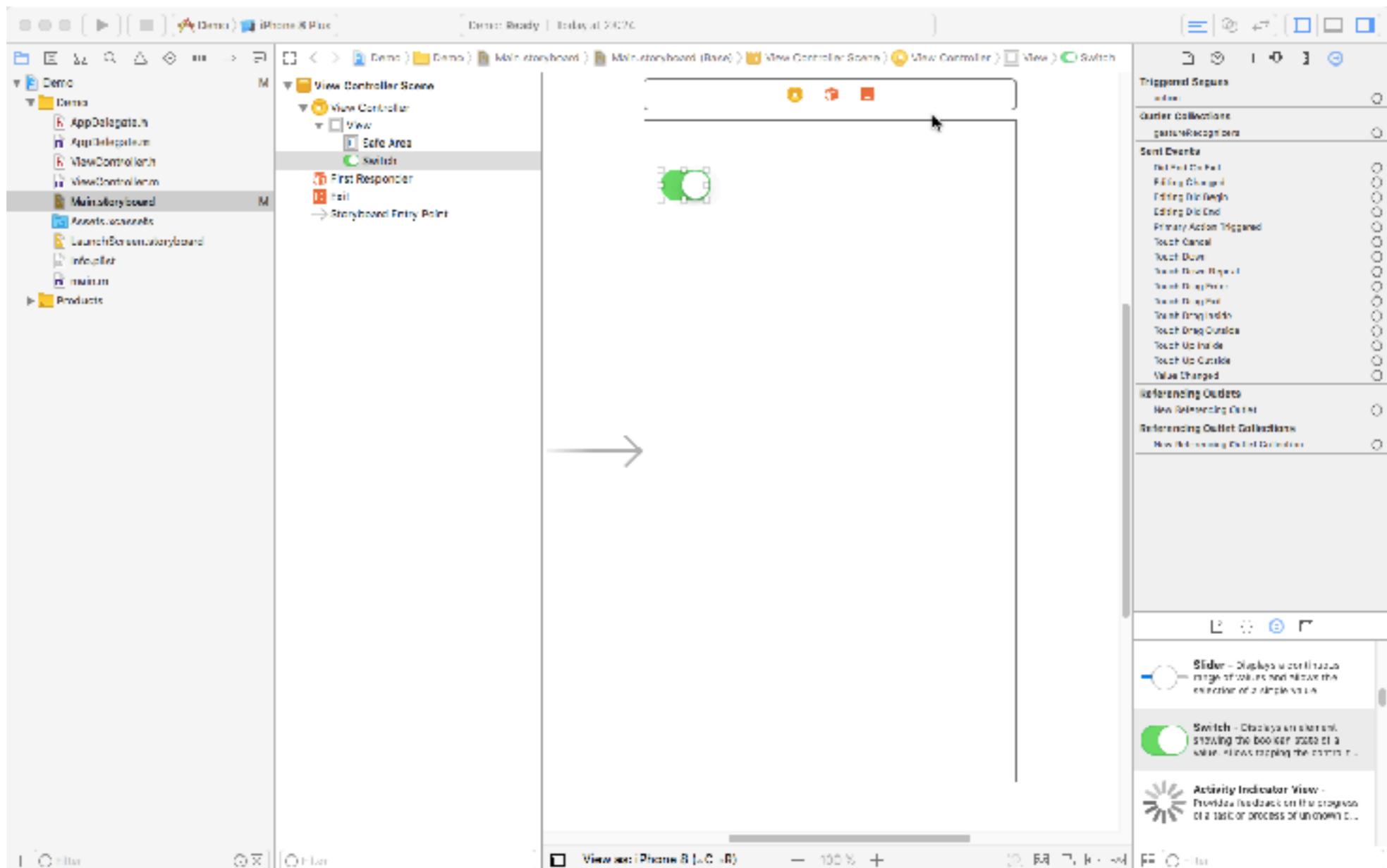
Démo Xcode 9

24



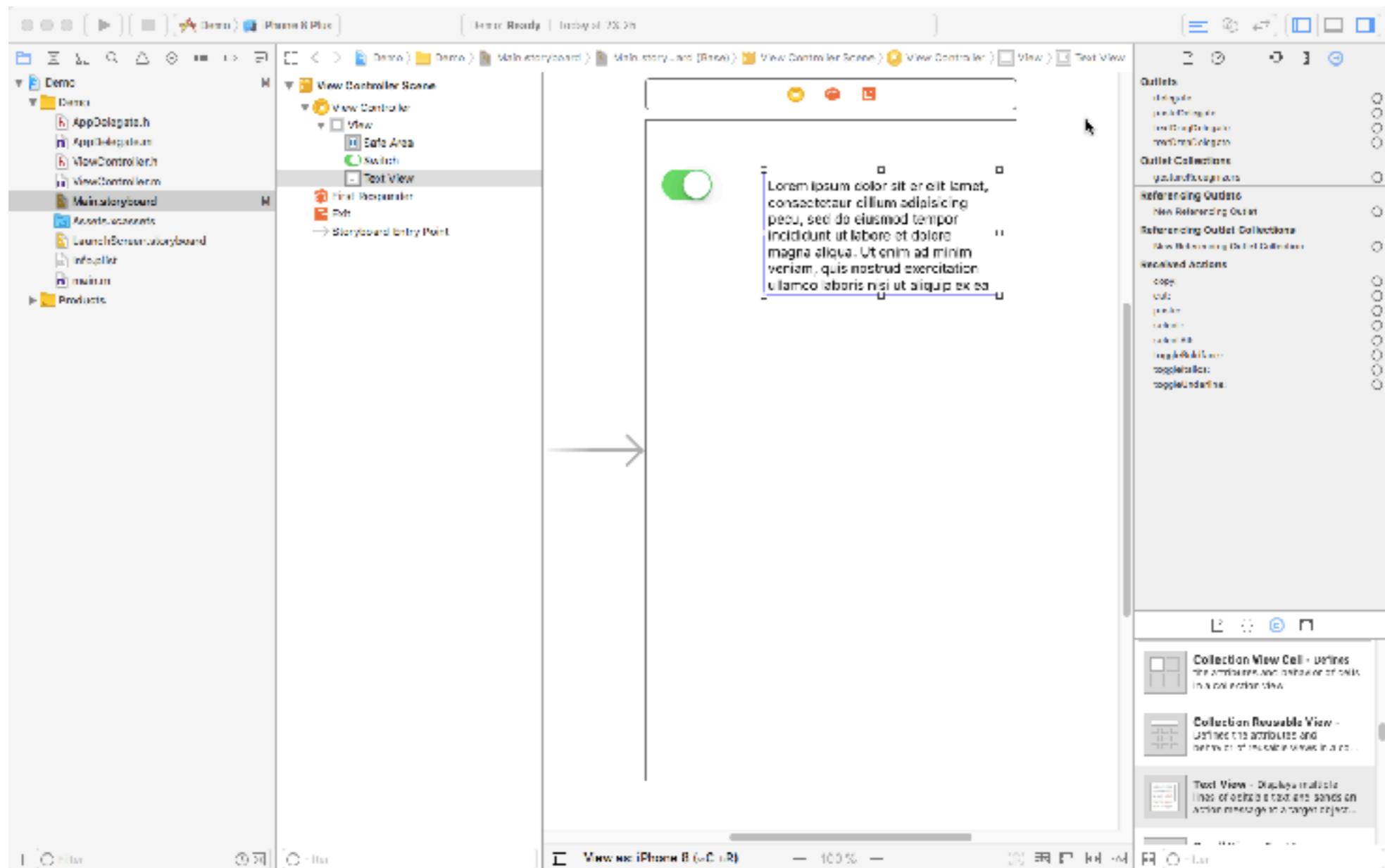
Démo Xcode 9

24



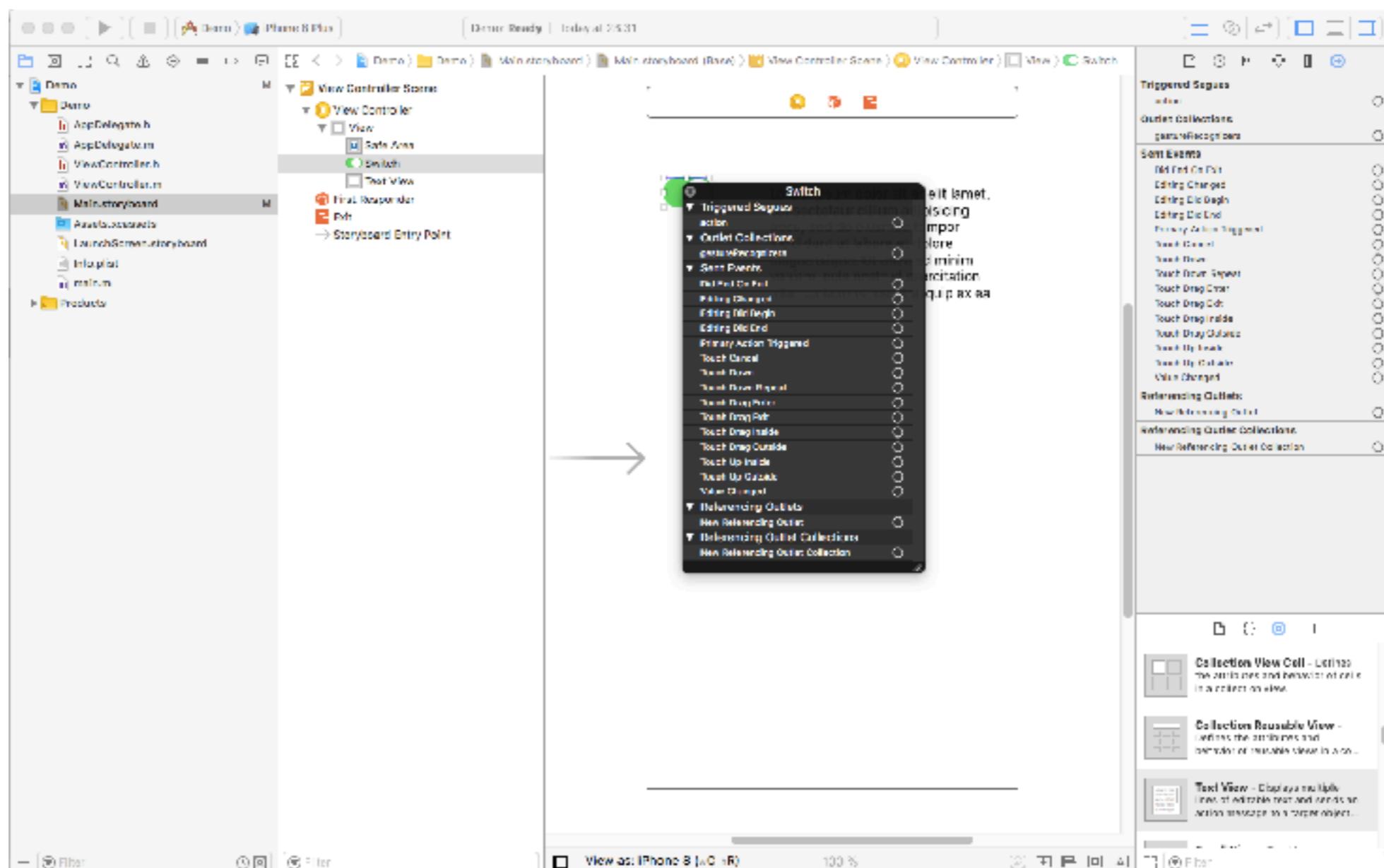
Démo Xcode 9

24



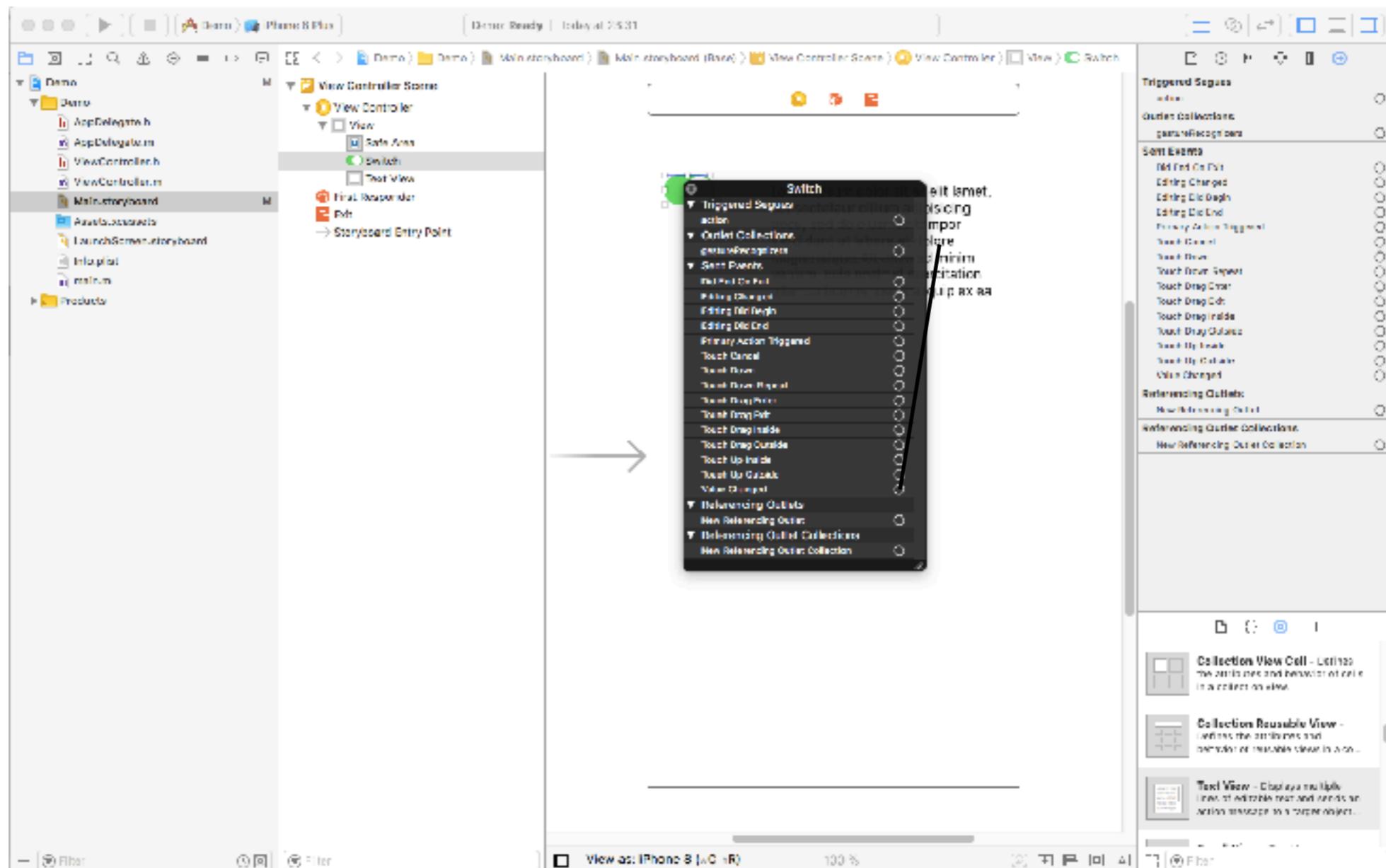
Démo Xcode 9

24



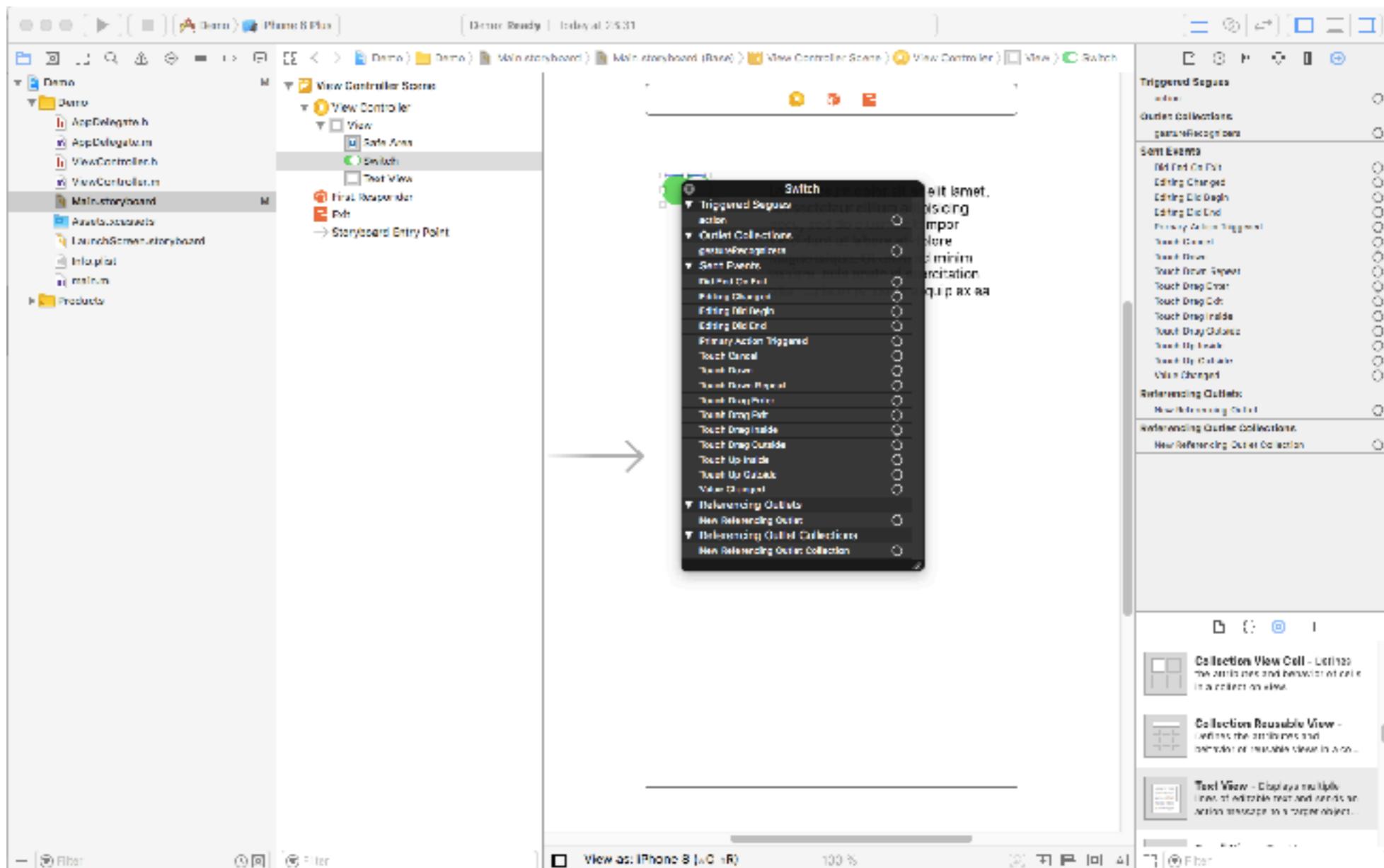
Démo Xcode 9

24



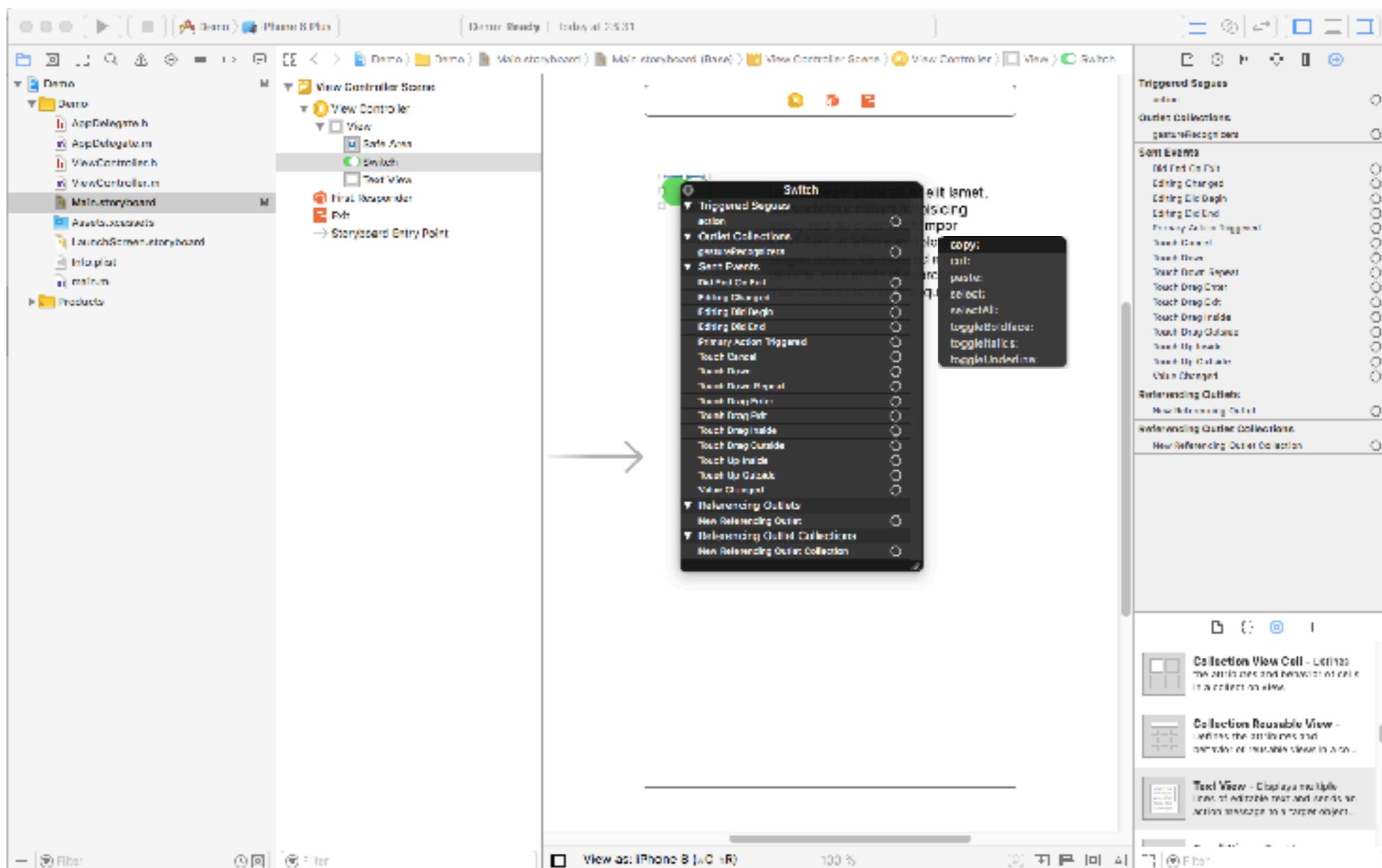
Démo Xcode 9

24



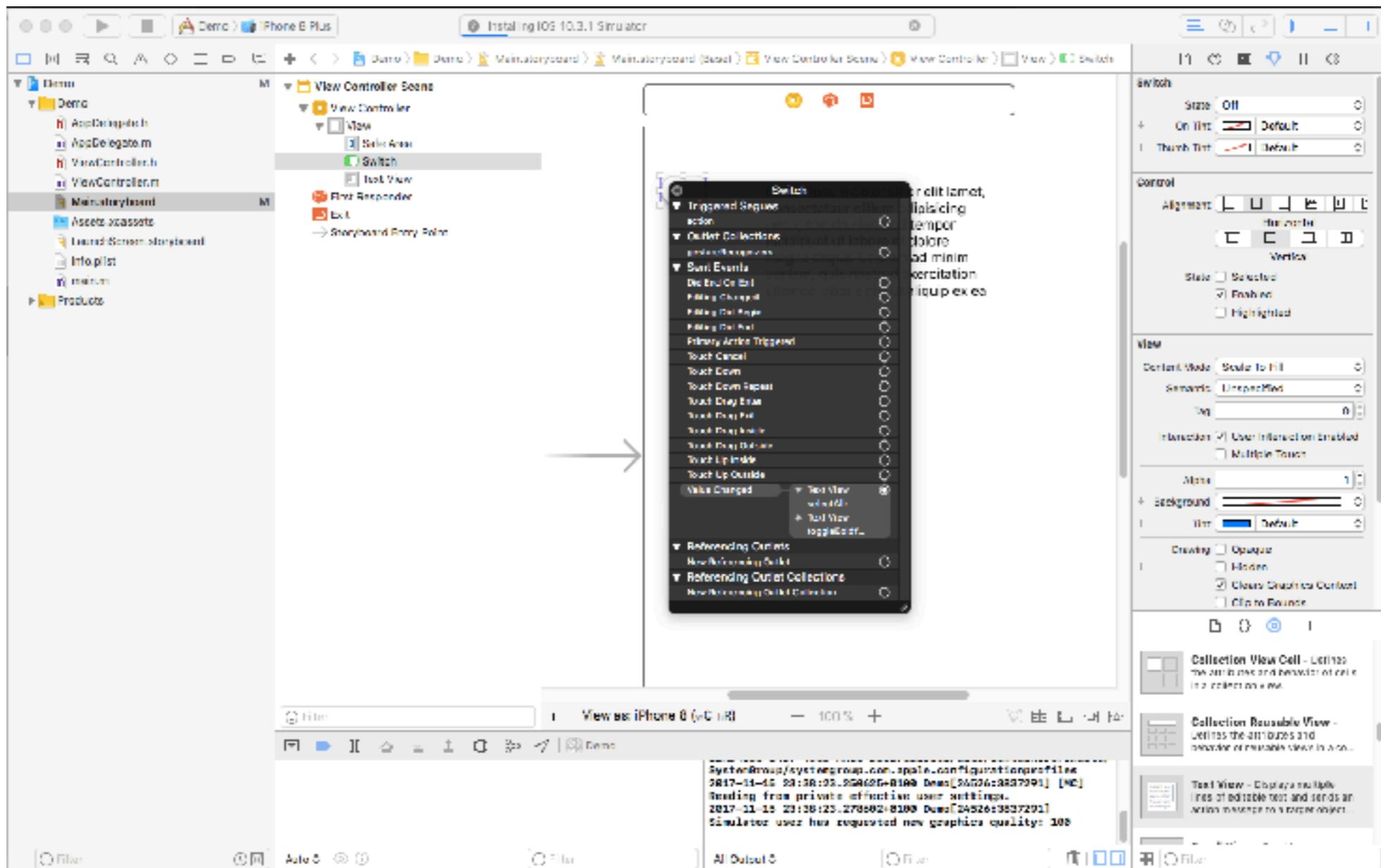
Démo Xcode 9

24



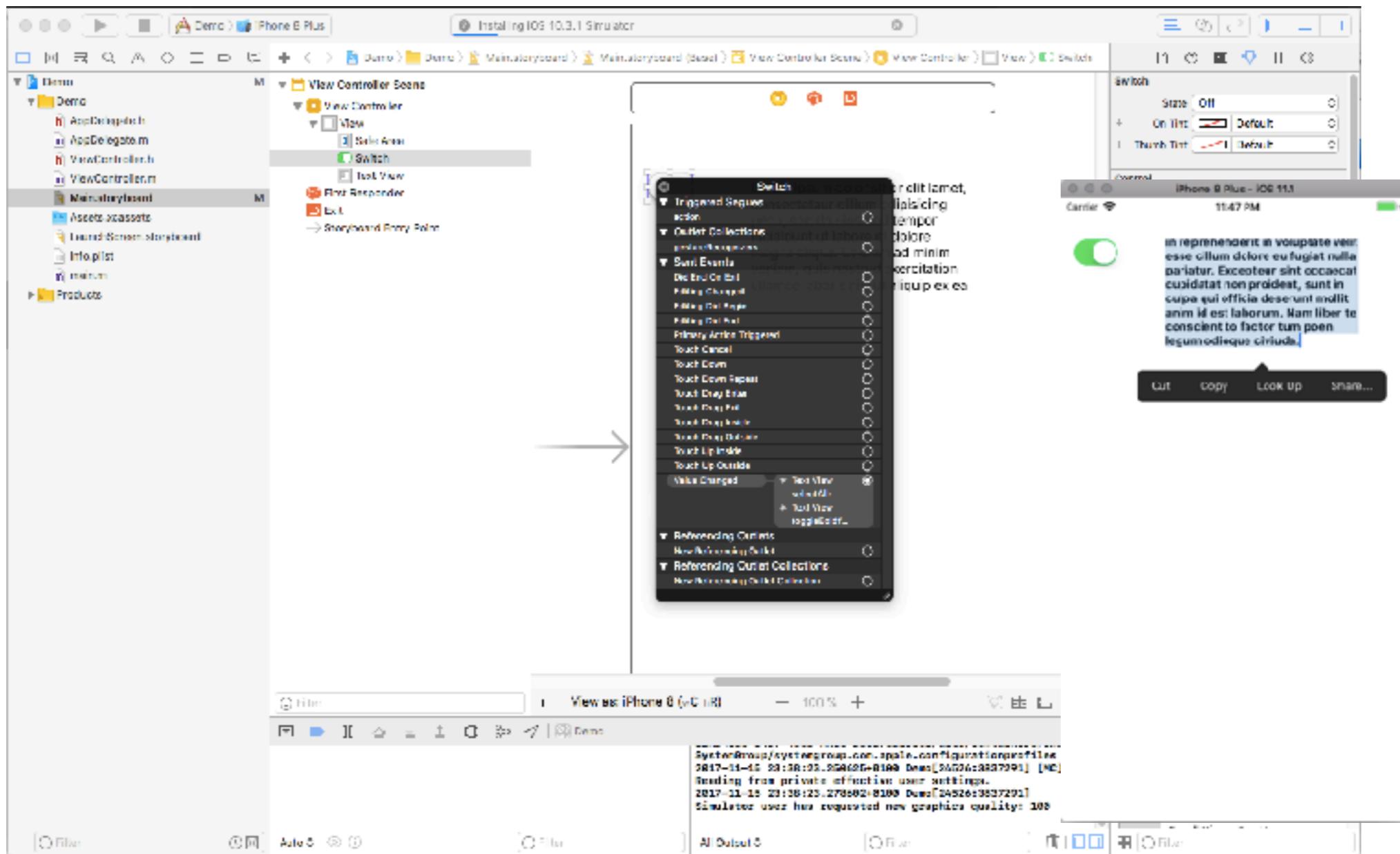
Démo Xcode 9

24



Démo Xcode 9

24



Et maintenant Swift

25

- De nombreux emprunts aux langages fonctionnels (Haskell, Ocaml...) et objets (Objective-C, Ruby, Python...)
 - Fonctions, clôtures, classes, énumérations, protocoles, structures, extensions
 - Typage fort, inférence de types, types options
 - pattern matching
 - Intégration dans le langage des principaux types du FoundationFramework (NSString, NSDictionary...)
 - Pas de pointeurs (pas de calcul arithmétique sur les références) !
 - Generic types, enumerations

Et maintenant Swift

26

- prévu pour coexister avec Objective-C
- gestion mémoire : ARC et des annotations spécifiques sous formes de mots clés (**weak** et **unowned**)

Swift 2.0

27

- Version OpenSource avec une distribution pour Linux !
- Le langage est complété pour être parfaitement interprétable avec Objective-C
- gestion des exceptions (mot clé **throws**). Gestion des exception par des **defer-do-try-catch** dont le sens est assez différents des équivalents Java !
- Protocol extensions (similaires aux catégories d'Objective-C)
- gros travail sur le typage des énumérations

Swift 2.0

28

- Extension du support du pattern matching **guard**, **for-in** ...
- gestion des versions d'APIs différentes :
#available() et **@available()**.
if #available(iOS 9, *){
 ...
}
else{
 ...
}

Swift 3

29

- des changements de conventions syntaxiques qui obligeront à quelques adaptations de code.
- suppression des opérateur de types ++, des boucles for de la forme langage C...
- Un outil de migration corrigera automatiquement un grand nombre d'erreur.
- gestionnaire de package standard.
- support Linux amélioré.
- Le playground disponible directement sur iOS.

Swift 4

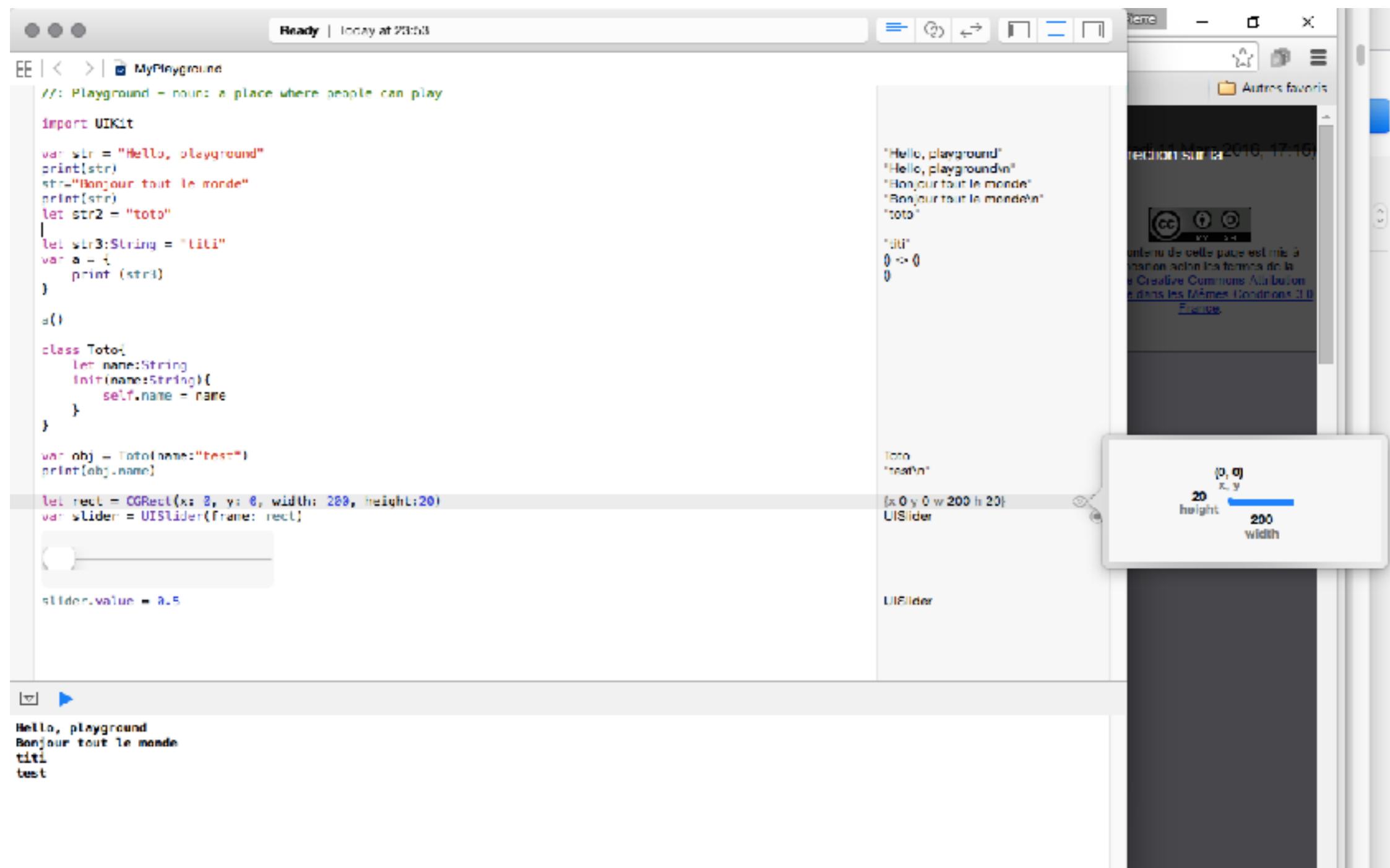
30

- moins en rupture que la version 3.
- Réimplantation des chaînes de caractères et des collections pour plus d'efficacité et de facilité d'écriture ajout de fonctionnalités .
- Nouveaux mécanisme de sérialisation JSON, plist.
- Mécanismes de détection de conformité (classes, protocols) étendus.
- Toujours pas de stabilisation des ABIs (repoussée de la version 3 à la version 4 puis maintenant la future versions 5).

le play ground

le play ground

31



Les applications

Les applications

32

- Le système met l'accent sur les applications (app-centric)

Les applications

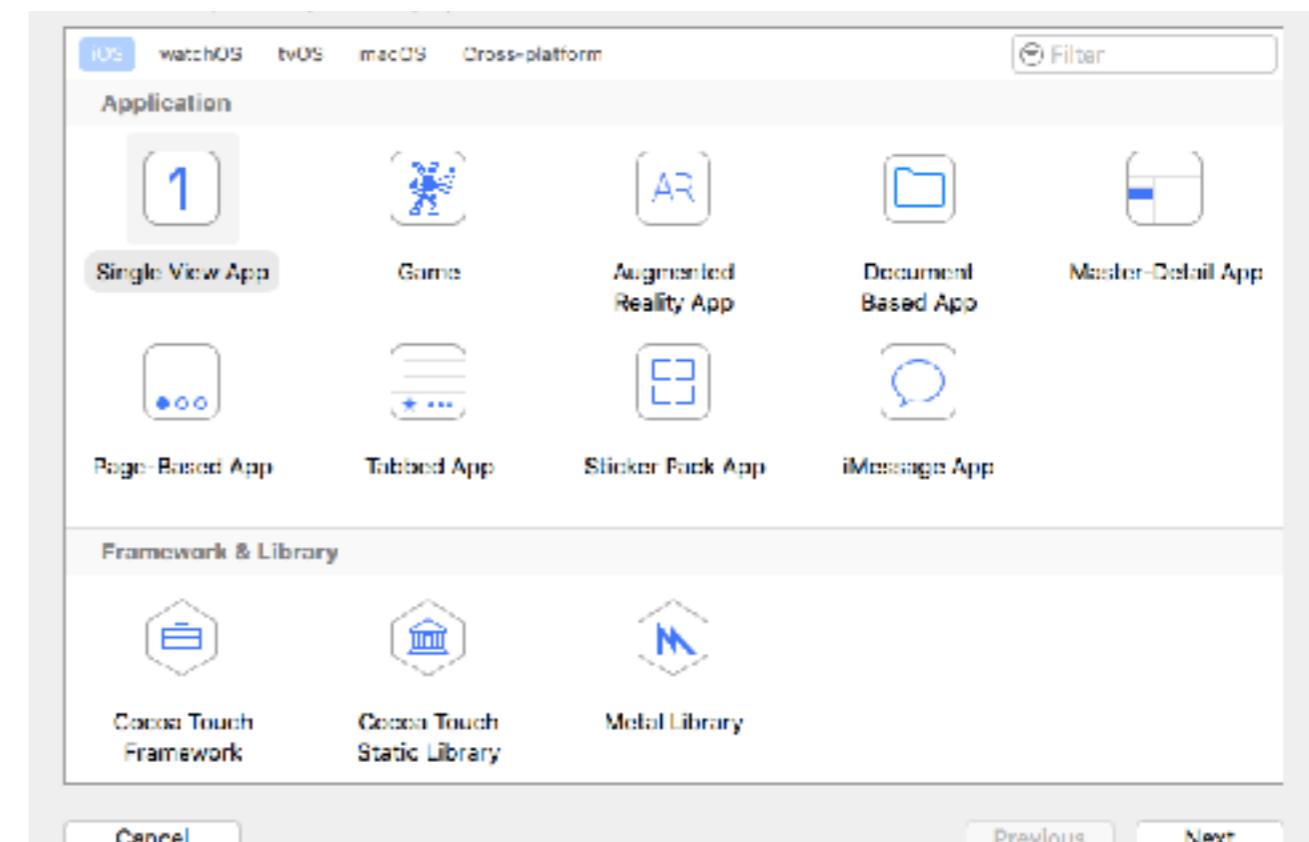
32

- Le système met l'accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d'applications :

Les applications

32

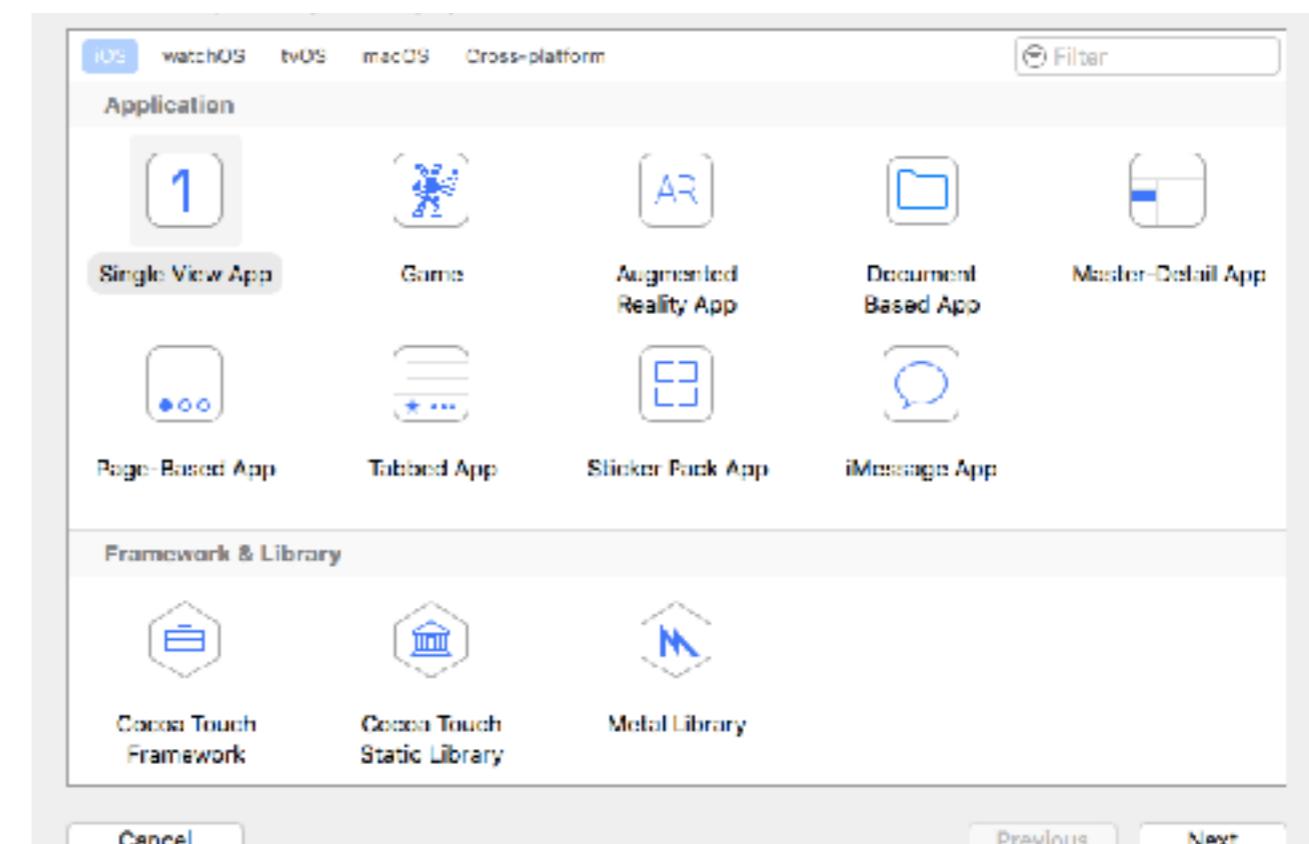
- Le système met l'accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d'applications :



Les applications

32

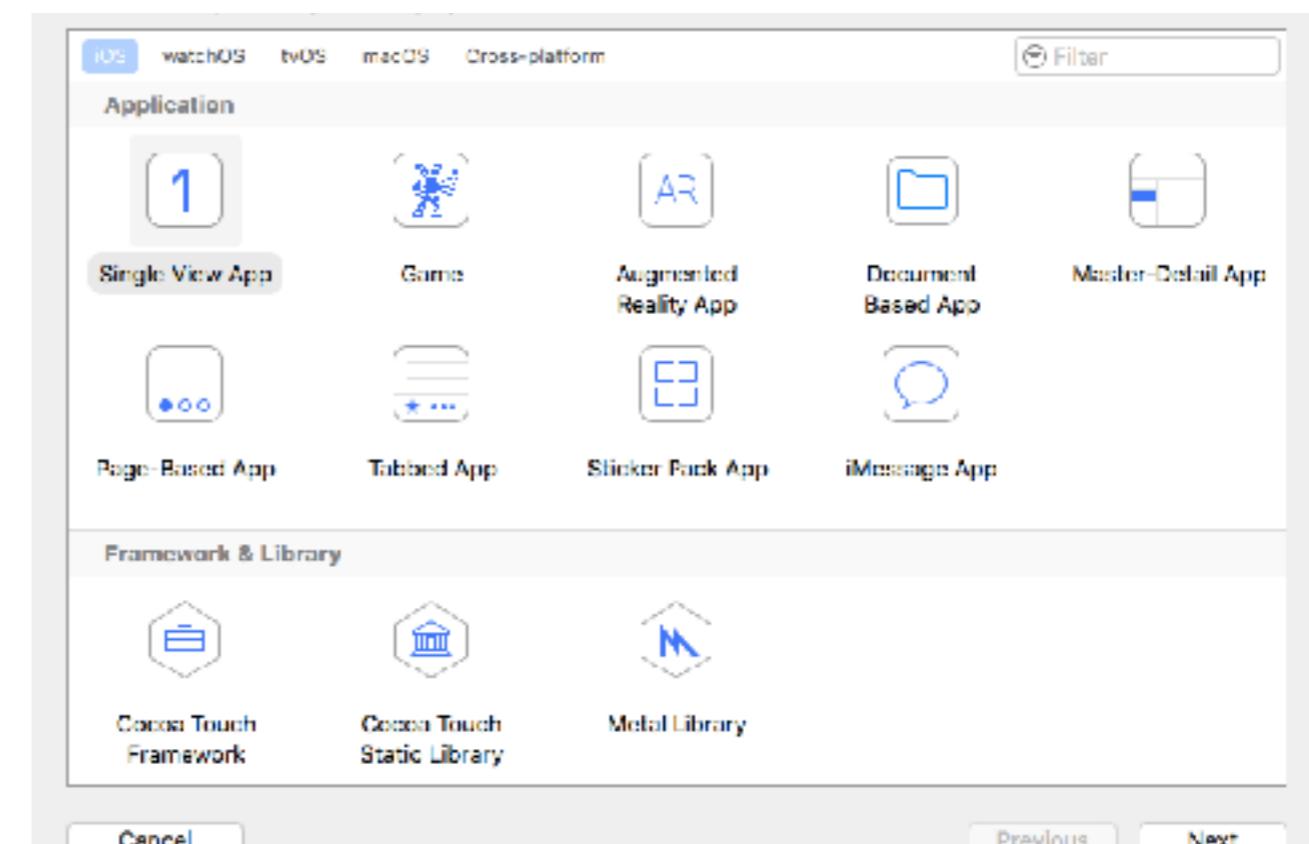
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps



Les applications

32

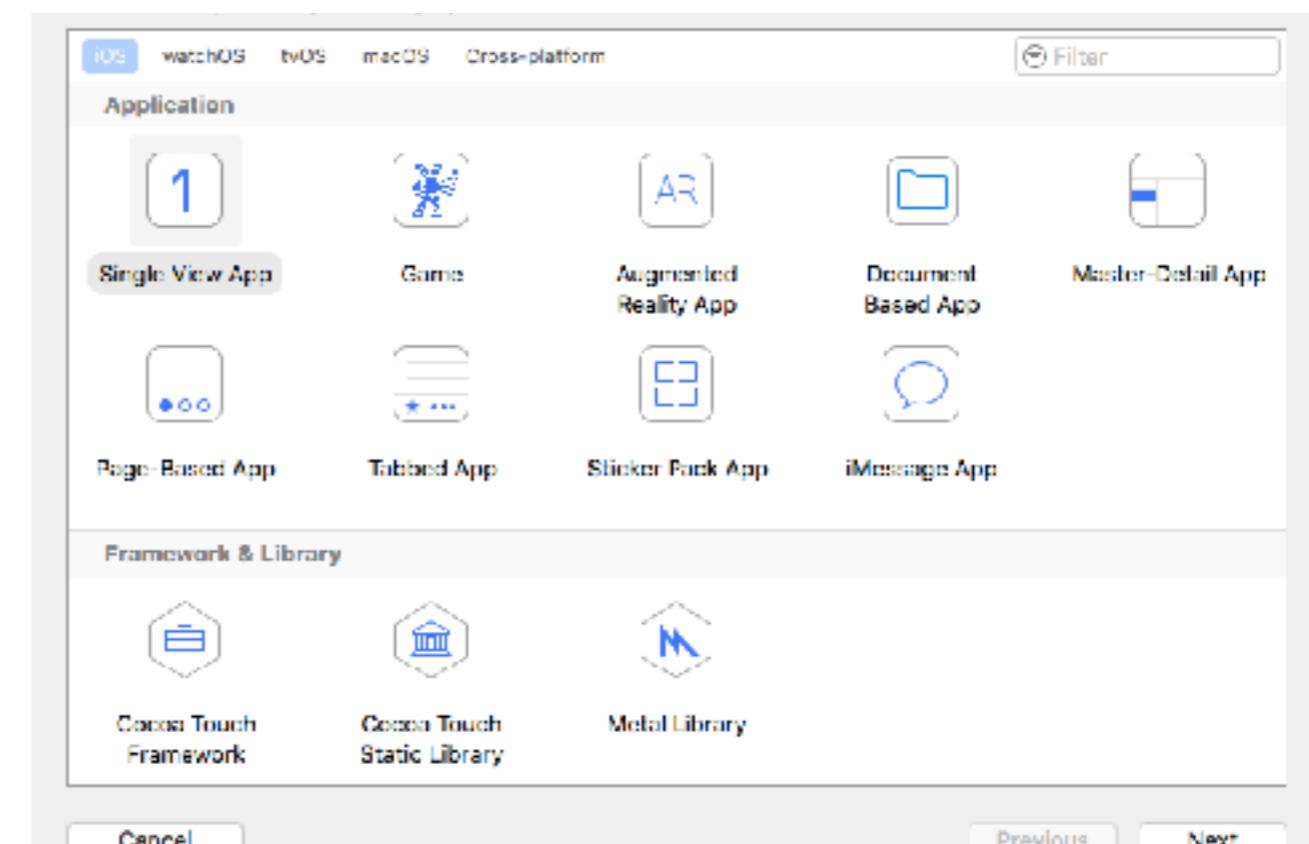
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps



Les applications

32

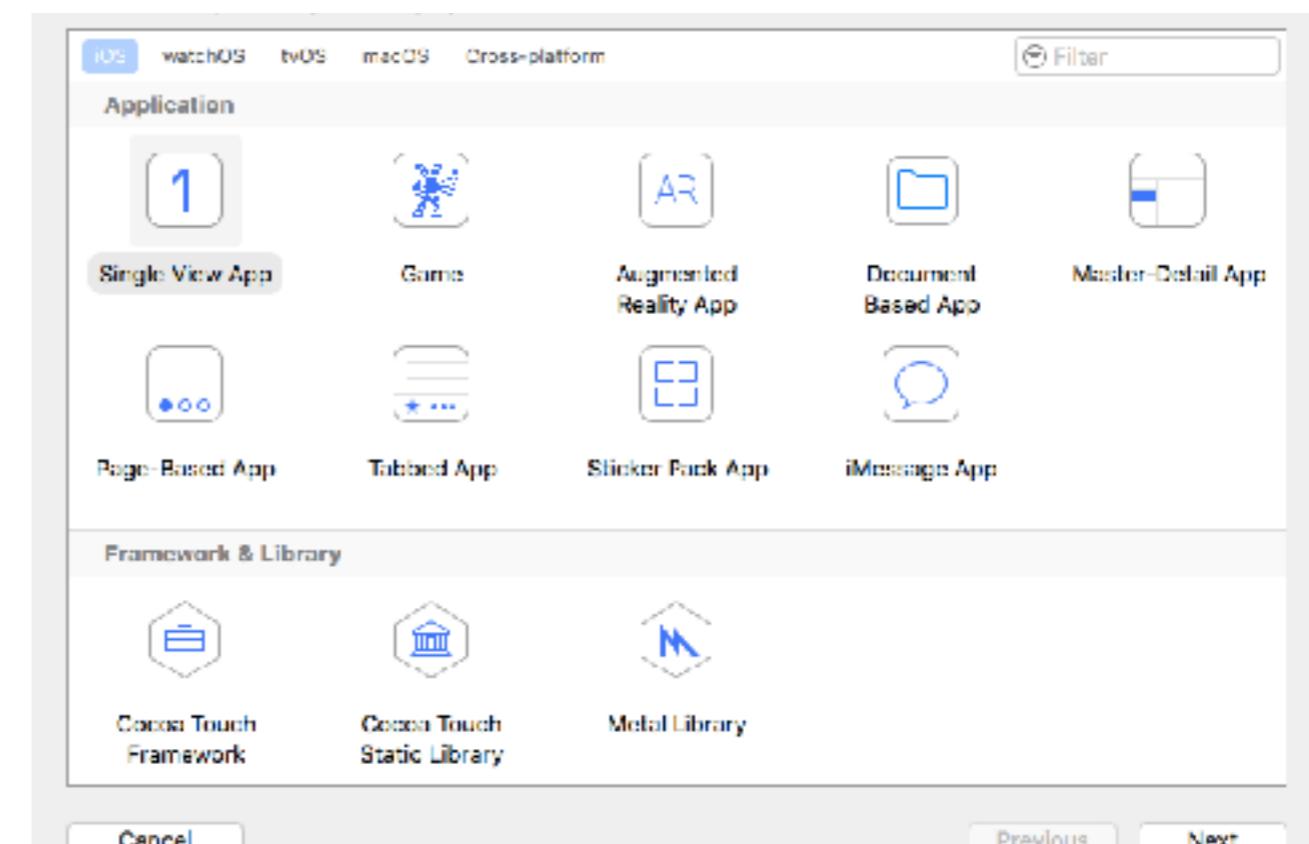
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps
 - Single View Apps



Les applications

32

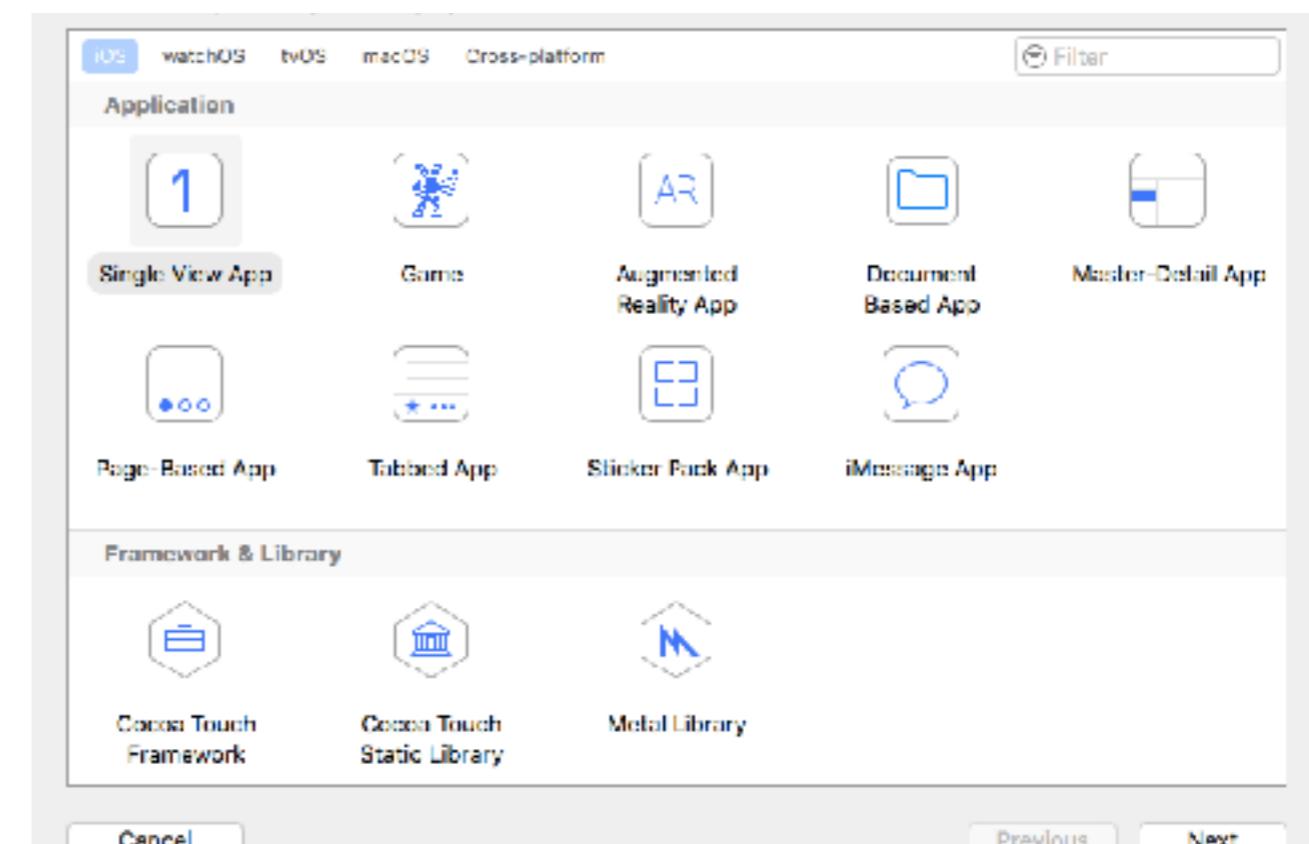
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps
 - Single View Apps
 - Tabbed Apps



Les applications

32

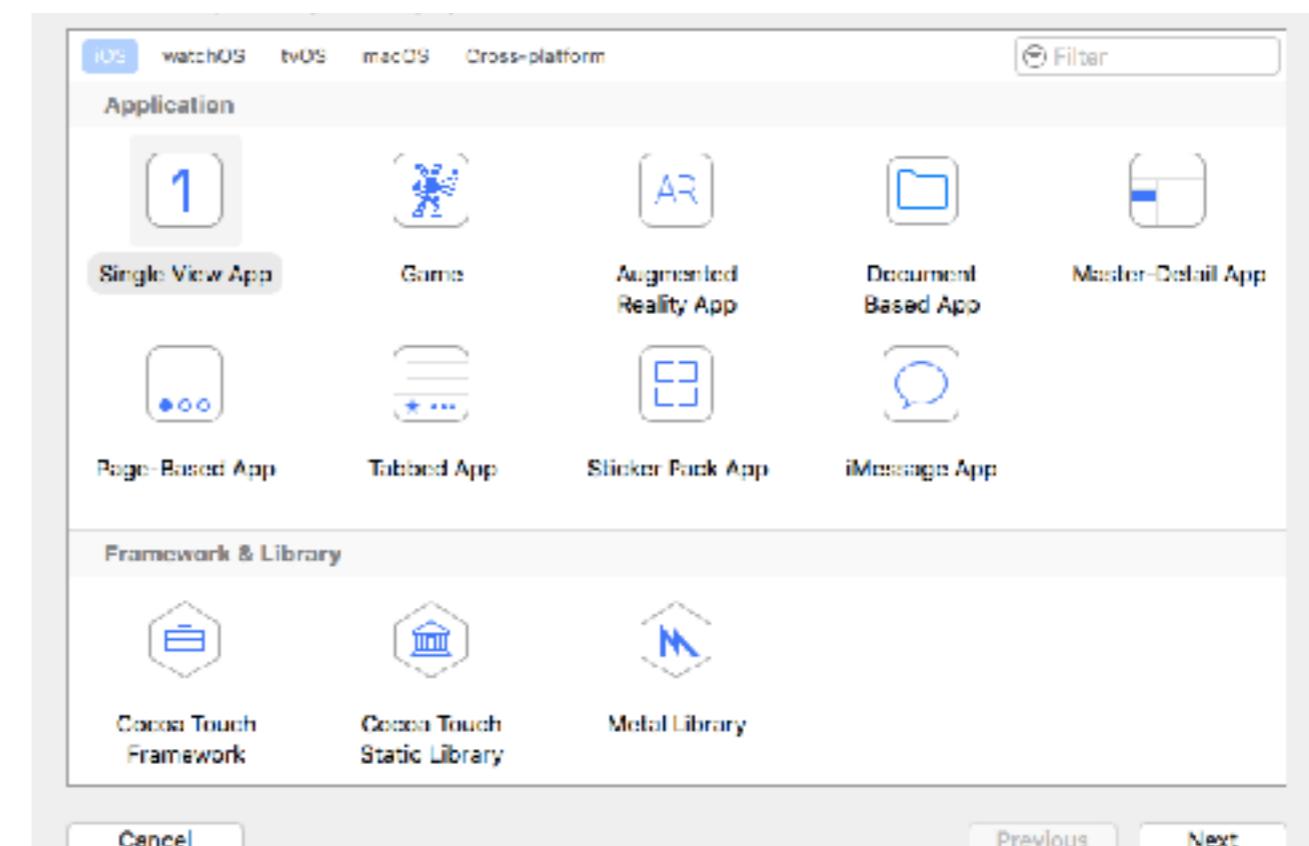
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps
 - Single View Apps
 - Tabbed Apps
 - Game



Les applications

32

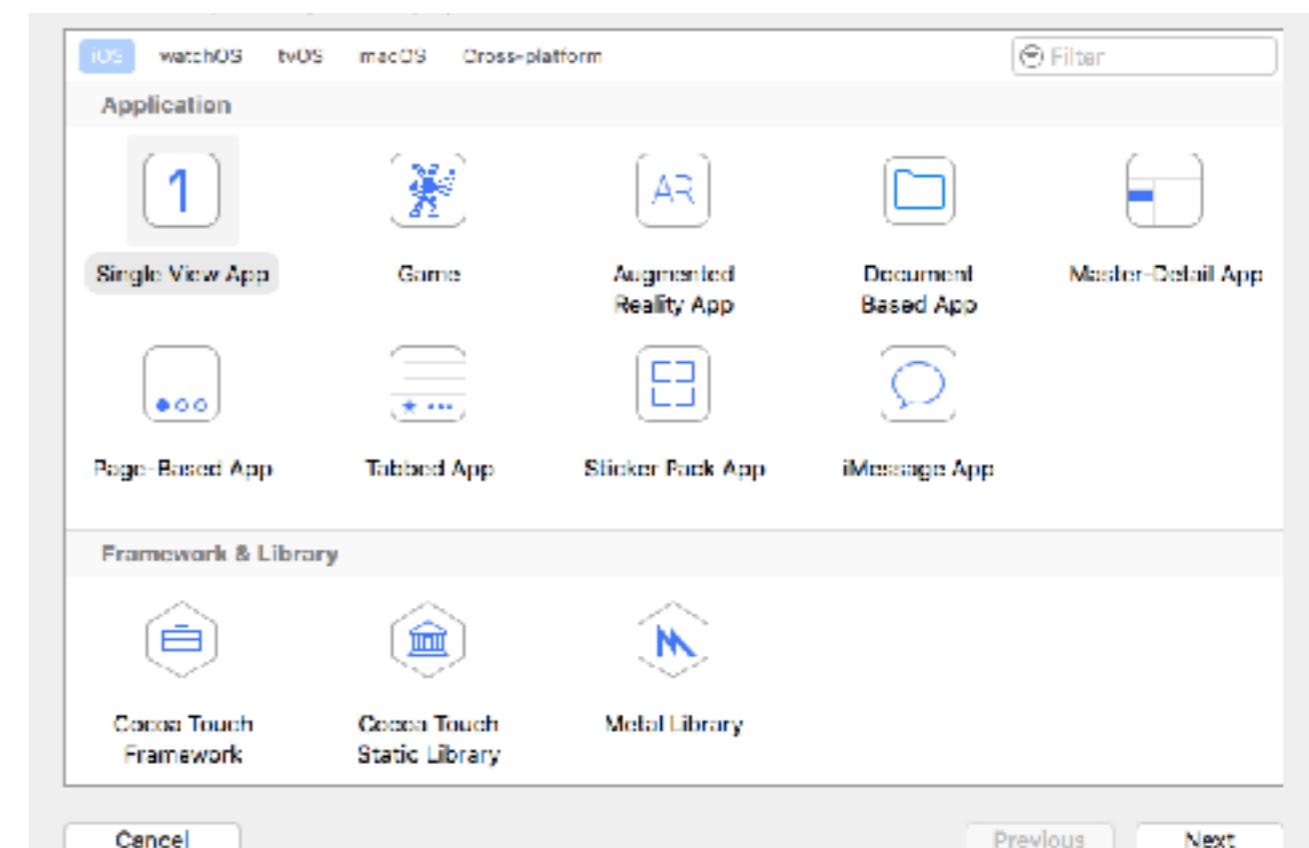
- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps
 - Single View Apps
 - Tabbed Apps
 - Game
 - AR App



Les applications

32

- Le système met l’accent sur les applications (app-centric)
- Une application : 1 tâche, les applications démarrent rapidement (splash screens). Différents types d’applications :
 - Master-Detail Apps
 - Page-Based Apps
 - Single View Apps
 - Tabbed Apps
 - Game
 - AR App
 - Sticker Pack App et iMessage App



Les applications

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4 <dict>
5   <key>appleId</key>
6   <string>████████████████████████████████████████████████████████████████████████████████████████████████████████████</string>
7   <key>artistId</key>
8   <integer>████████████████████████████████████████████████████████████████████████████████████████████████████████</integer>
9   <key>artistName</key>
10  <string>████████████████████████████████████████████████████████████████████████████████████████████████████</string>
11  <key>bundleId</key>
12  <string>████████████████████████████████████████████████████████████████████████████████████████████████</string>
13  <key>bundleShortVersionString</key>
14  <string>1.4.B&lt;/string>
15  <key>bundleVersion</key>
16  <string>1.4.6</string>
17  <key>copyright</key>
18  <string>████████████████████████████████████████████████████████████████████████████████████████████████</string>
19  <key>fileExtension</key>
20  <string>.app</string>
21  <key>genre</key>
22  <string>Puzzle</string>
23  <key>isAdSupported</key>
24  <integer>0</integer>
25  <key>isFree</key>
26  <integer>1</integer>
27  <key>itemCount</key>
28  <integer>1</integer>
29  <key>interlace</key>
30  <string>████████████████████████████████████████████████████████████████████████████████████████████</string>
31  <key>kind</key>
32  <string>software</string>
33  <key>languageName</key>
34  <string>████████████████████████████████████████████████████████████████████████████████████████</string>
35  <key>listName</key>
36  <string>████████████████████████████████████████████████████████████████████████████████████</string>
37  <key>parentTitle</key>
38  <date>████████████████████████████████████████████████████████████████████████████████████</date>
39  <key>rating</key>
40  <double>
41    <key>comScore</key>
42    <string>████████████████████████████████████████████████████████████████████████████████</string>
43    <key>label</key>
44    <string>4+</string>
45    <key>rank</key>
46    <integer>1000</integer>
47    <key>system</key>
48    <string>iTunes Games</string>
49  </double>
50 </dict>
```

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode

appleId	String	[REDACTED]
artistId	Number	[REDACTED]
artistName	String	[REDACTED]
New_item	String	[REDACTED]
bundleShortVersionString	String	1.4.6
bundleVersion	String	1.4.6
copyright	String	[REDACTED]
dmvCommentsNumber	Number	0
fileExtension	String	.app
genre	String	Productivité
genreId	Number	6007
itemId	Number	[REDACTED]
itemName	String	[REDACTED]
kind	String	software
playlistArtistName	String	[REDACTED]
playlistName	String	[REDACTED]
purchaseDate	Date	[REDACTED]
rating	Dictionary	(4 items)
content	String	[REDACTED]
label	String	4+
rank	Number	100
system	String	iTunes-games
releaseDate	String	[REDACTED]
z	Number	[REDACTED]
softwareIcon57x57URL	String	[REDACTED]
softwareIconNeedsShine	Boolean	NO
softwareSupportedDeviceIds	Array	(1 item)
Item 0	Number	1
softwareRevisionBundled	String	[REDACTED]

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode
 - iTunesArtwork (image 512x512)

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode
 - iTunesArtwork (image 512x512)
 - _CodeSignature/CodeResources : PropertyList contenant la signature de chaque fichier de l'archive

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode
 - iTunesArtwork (image 512x512)
 - _CodeSignature/CodeResources : PropertyList contenant la signature de chaque fichier de l'archive
 - Le répertoire Payload contient l'application MacOS X (dossier Bundle avec l'extension .app)

Les applications

33

- Un format propriétaire de distribution sous iTunes (archive zip) le format ipa contenant l'ensemble des fichiers :
 - iTunesMetadata.plist fichier XML (DTD Parameter List) Editeur dédié dans XCode
 - iTunesArtwork (image 512x512)
 - _CodeSignature/CodeResources : PropertyList contenant la signature de chaque fichier de l'archive
 - Le répertoire Payload contient l'application MacOS X (dossier Bundle avec l'extension .app)
 - Lien vers le fichier de signatures

Les applications

34

- Application.app contient un bundle applicatif de MacOS X
 - Info.plist : contient des informations (property list) sur l’application
 - Le code exécutable (MachO code format) même nom que le dossier sans le .app (binaire signé)
 - fichiers ressources : images, sons, videos, templates... dont Default.png
 - les dossiers .lproj : contenant les fichiers .nib, les chaines de caractères à localiser...

Les applications

35

```
MyApp.app/
    Info.plist
    MyApp
    Default.png
    Icon.png
    Hand.png
    MainWindow.nib
    MyAppViewController.nib
    WaterSounds/
        Water1.aiff
        Water2.aiff
    en.lproj/
        CustomView.nib
        bird.png
        Bye.txt
        Localizable.strings
    jp.lproj/
        CustomView.nib
        bird.png
        Bye.txt
        Localizable.strings
```

Les applications

36

- Un seul utilisateur UNIX : l'utilisateur mobile
- Les applications sont installées dans /var/mobile/
Applications/....
- Chaque application est isolée dans son propre dossier
d'installation qui est son répertoire de travail. Pas de
dossier utilisateur accessible à tous, pas de partage facile
de données, elles sont cantonnées à l'application

Appli.app/
Documents/
Library/
Cache, Cookies,
Preferences

iTunesArtWork
iTunesMetadata.plist
tmp/

Les applications

37

- Les applications sont orientées par l'IHM. L'information doit-être succincte claire et structurée en fonction de l'importance de haut en bas
- Éviter au maximum la saisie par l'utilisateur
- Un seul mécanisme pour quitter : le bouton «Home»
- L'application doit-être prête à quitter n'importe quand (Home, appel téléphonique, multitâche, notifications...)
- Sauvegarde de l'état au moment où l'on arrête l'application



Les applications

38

- C'est un processus UNIX écrit dans un langage proche de C :

Les applications

38

- C'est un processus UNIX écrit dans un langage proche de C :

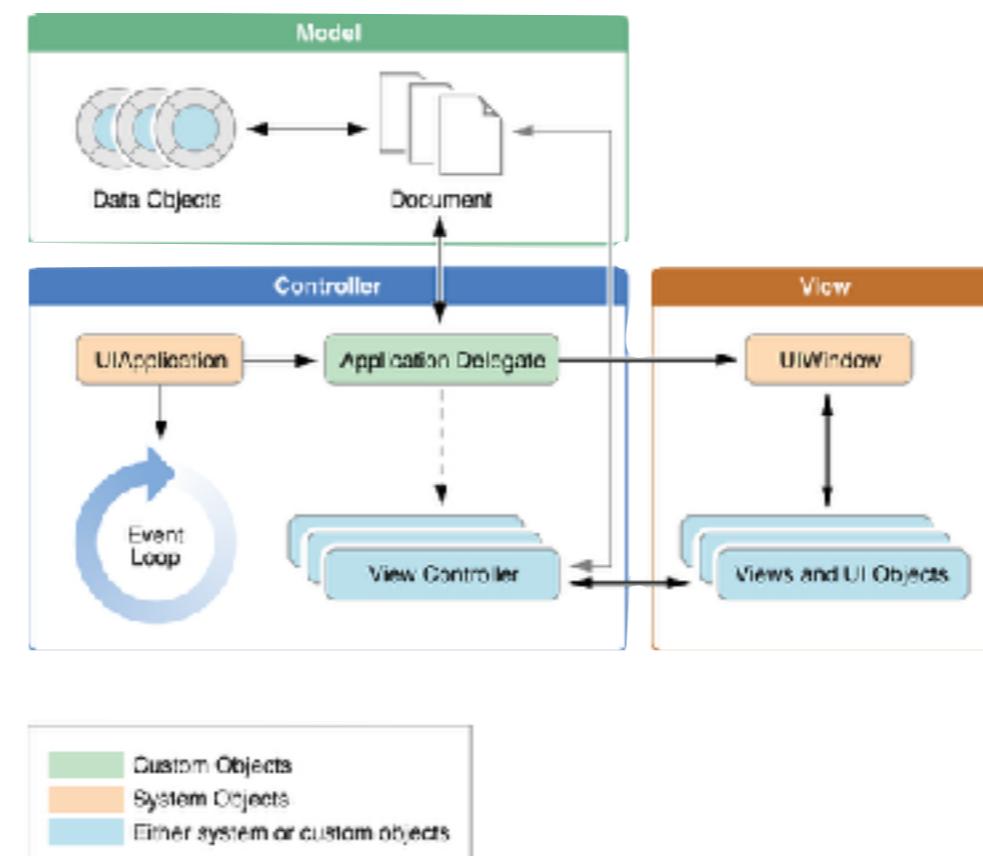
```
1 int main(int argc, char *argv[])
2 {
3     @autoreleasepool {
4         return UIApplicationMain(argc, argv, nil
5                             , NSStringFromClass([NFA022_AppDelegate class]));
6 }
```

Les applications

38

- C'est un processus UNIX écrit dans un langage proche de C :

```
1 int main(int argc, char *argv[])
2 {
3     @autoreleasepool {
4         return UIApplicationMain(argc, argv, nil
5                             , NSStringFromClass([NFA022_AppDelegate class]));
6 }
```

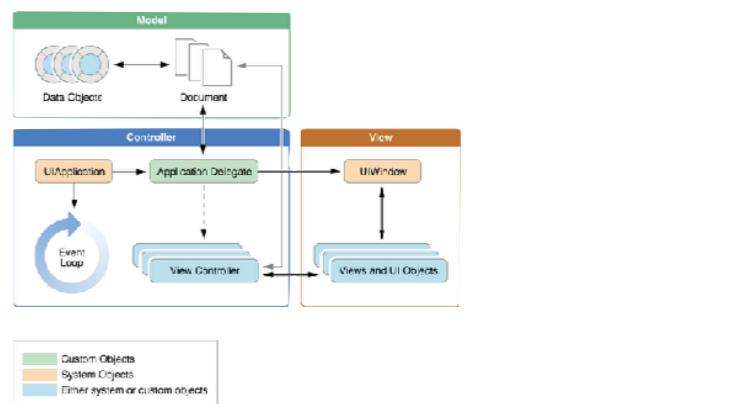


Les applications

38

- C'est un processus UNIX écrit dans un langage proche de C :

```
1 int main(int argc, char *argv[])
2 {
3     @autoreleasepool {
4         return UIApplicationMain(argc, argv, nil
5                             , NSStringFromClass([NFA022_AppDelegate class]));
6     }
}
```



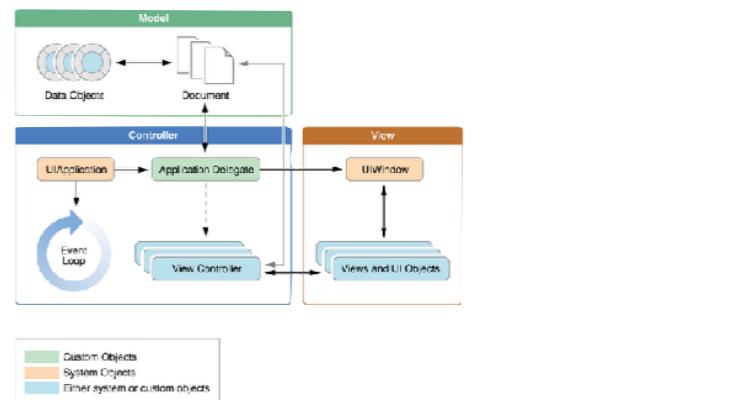
- Cycle de vie d'une application : implanté par un délégué

Les applications

38

- C'est un processus UNIX écrit dans un langage proche de C :

```
1 int main(int argc, char *argv[])
2 {
3     @autoreleasepool {
4         return UIApplicationMain(argc, argv, nil
5             , NSStringFromClass([NFA022_AppDelegate class]));
6     }
}
```

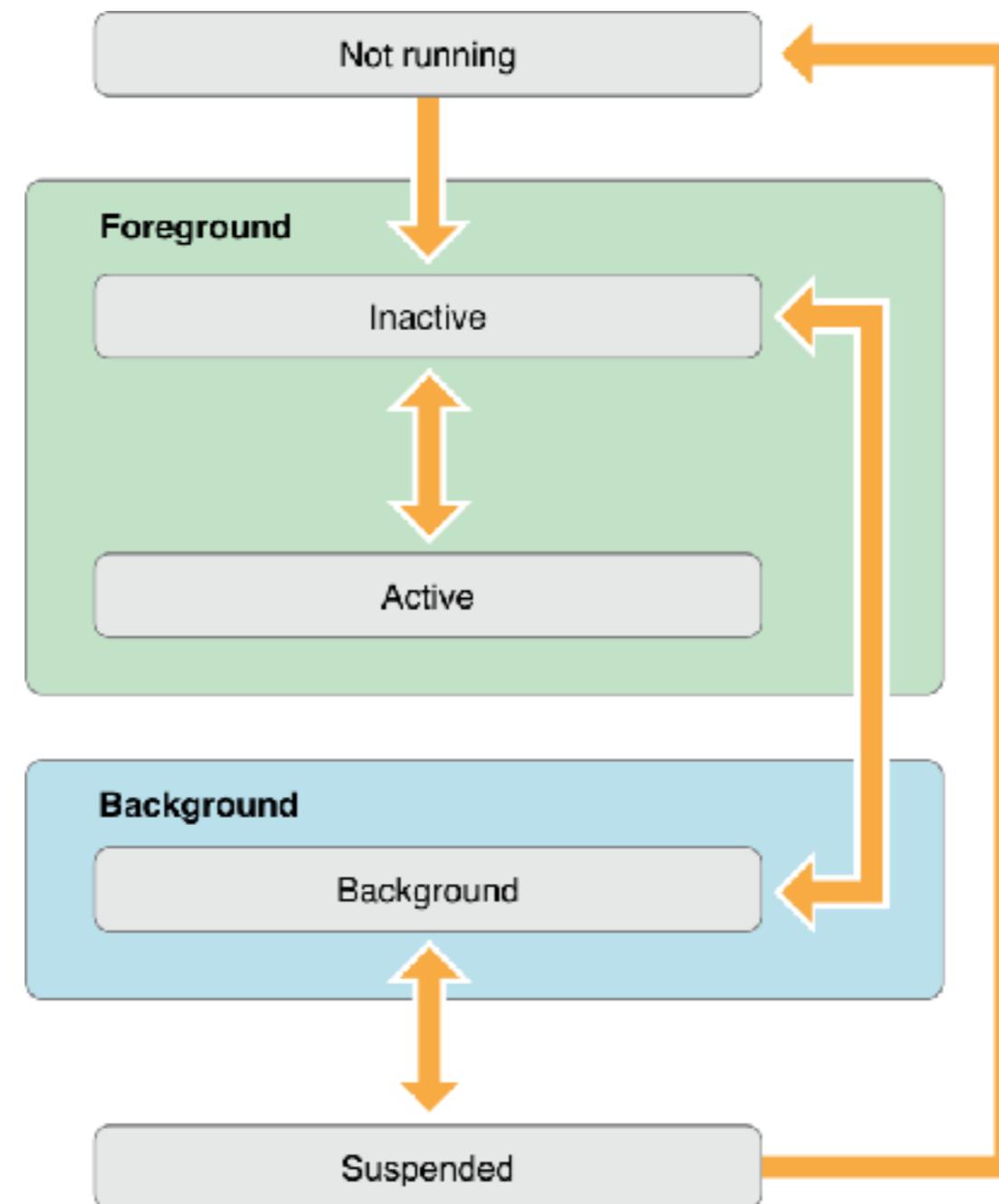


- Cycle de vie d'une application : implanté par un délégué

```
1 - (BOOL)application:(UIApplication *)application
2     didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
3 - (void)applicationWillResignActive:(UIApplication *)application
4 - (void)applicationDidEnterBackground:(UIApplication *)application
5 - (void)applicationWillEnterForeground:(UIApplication *)application
6 - (void)applicationDidBecomeActive:(UIApplication *)application
7 - (void)applicationWillTerminate:(UIApplication *)application
```

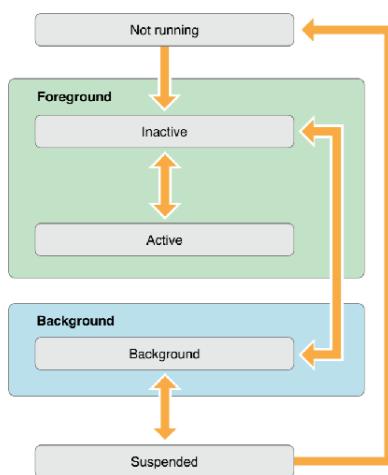
Cycle de vie des applications

39



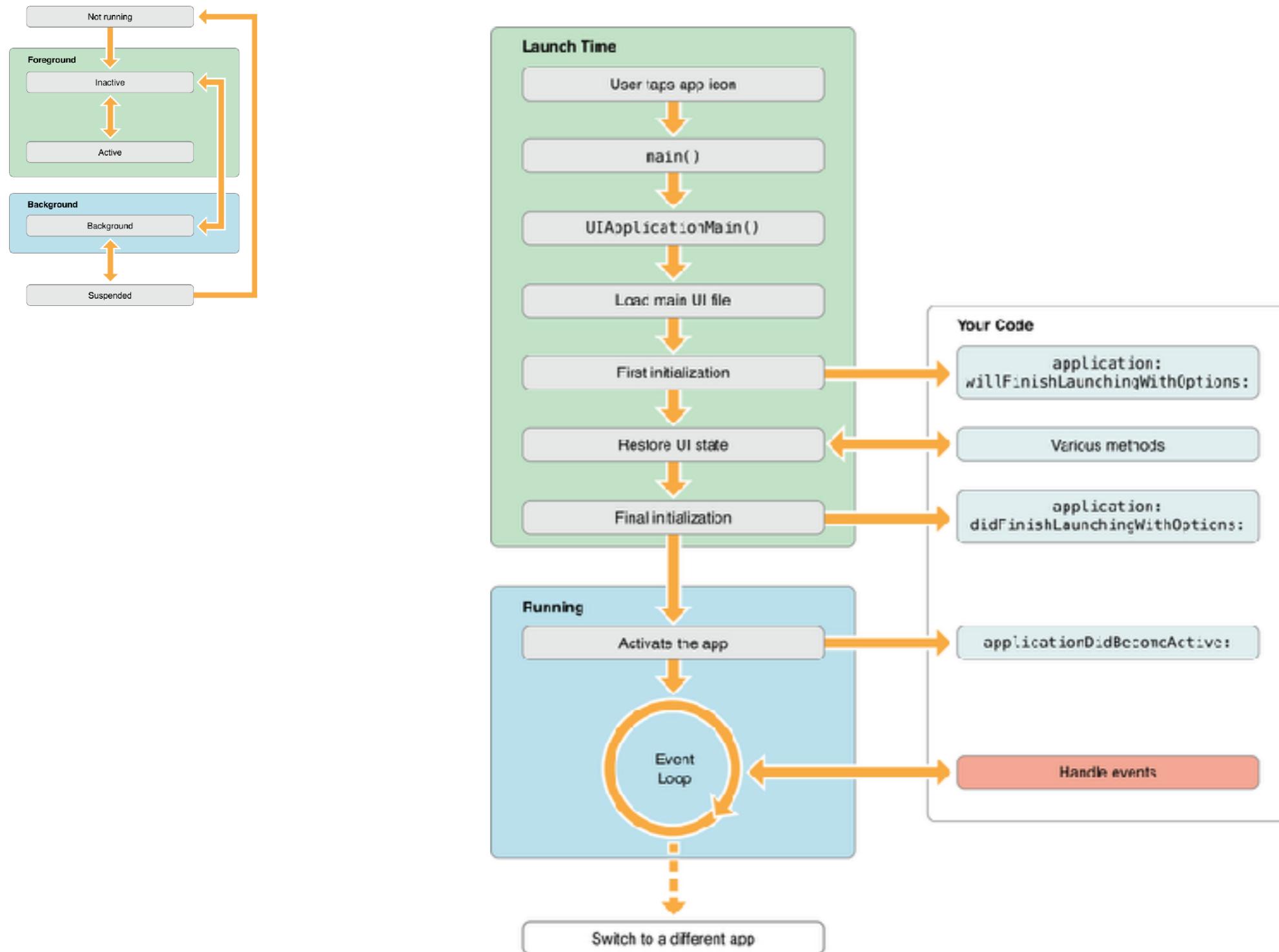
Cycle de vie des applications

39



Cycle de vie des applications

39



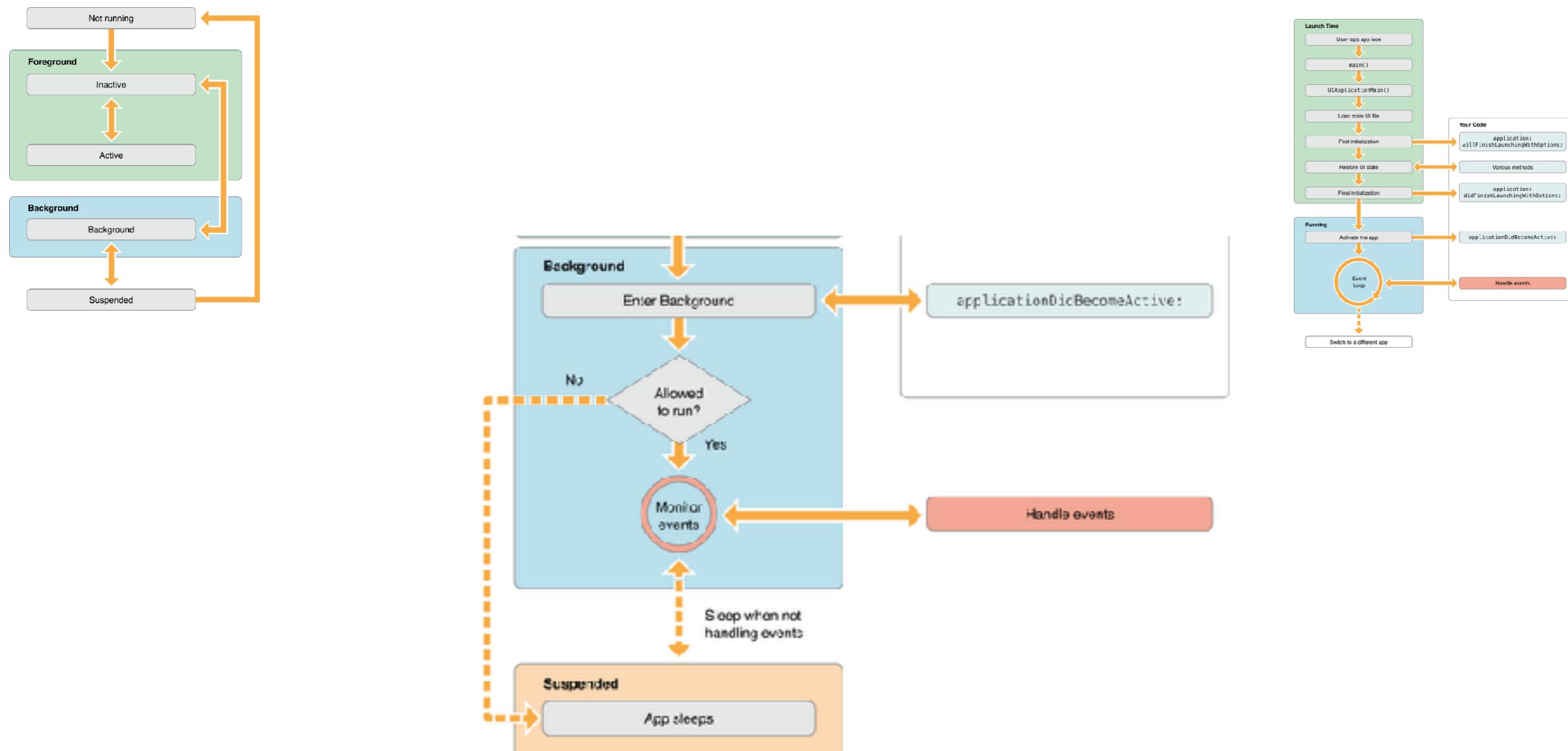
Cycle de vie des applications

39



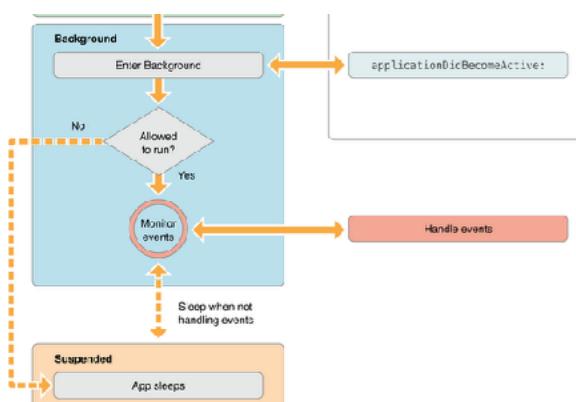
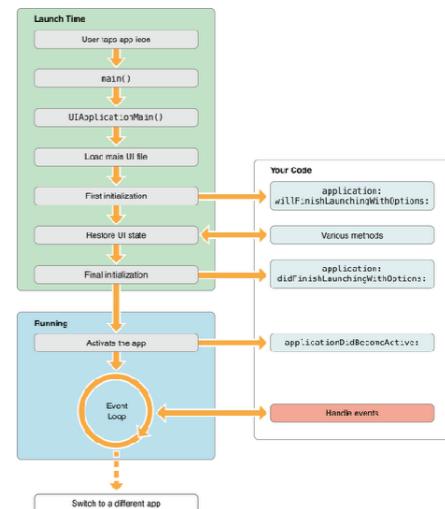
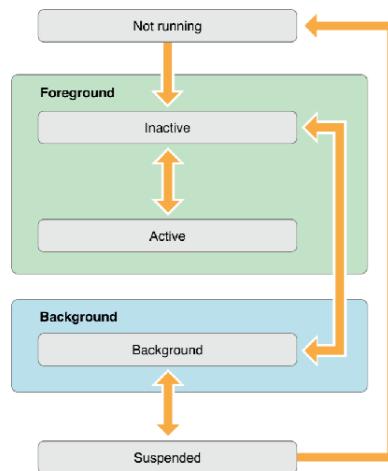
Cycle de vie des applications

39



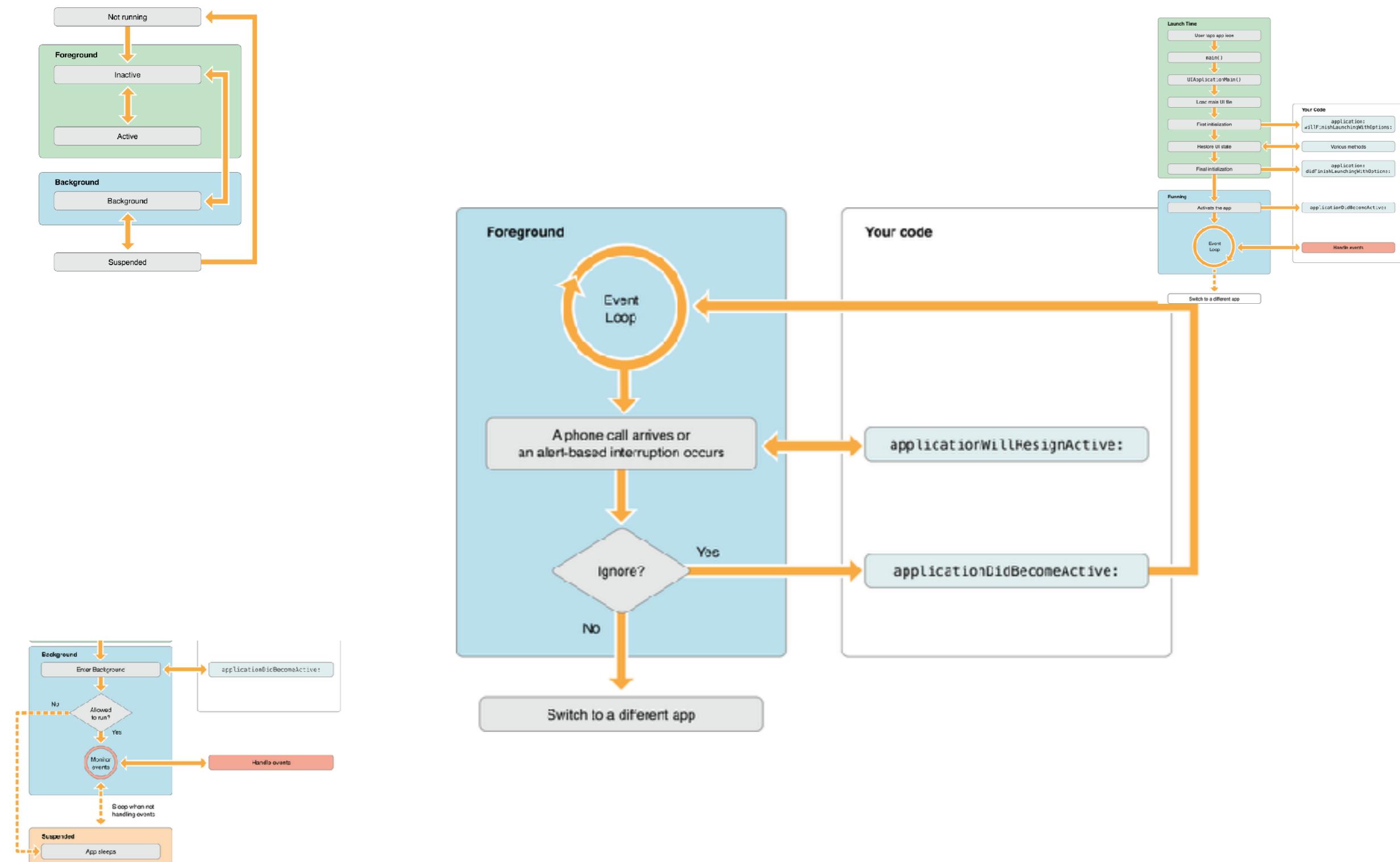
Cycle de vie des applications

39



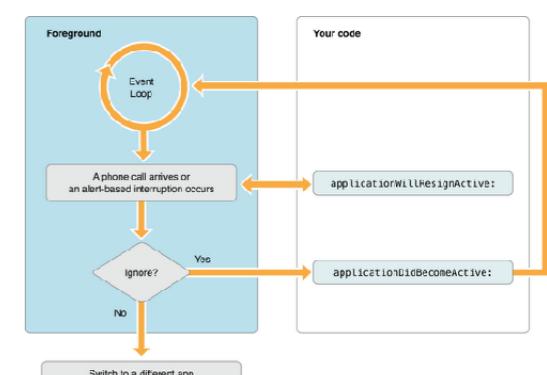
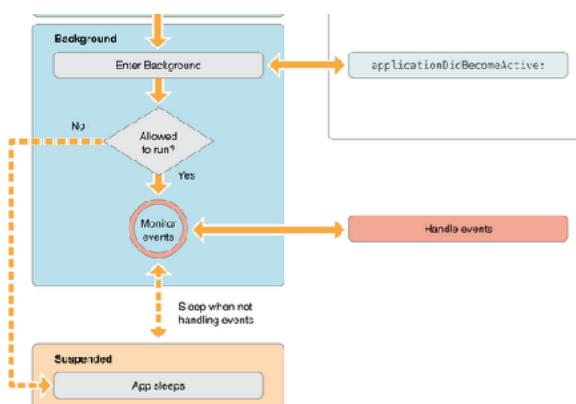
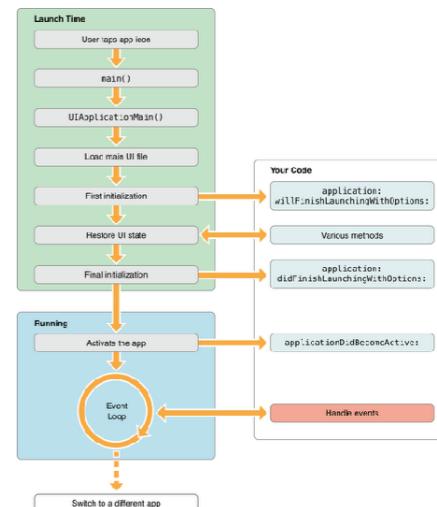
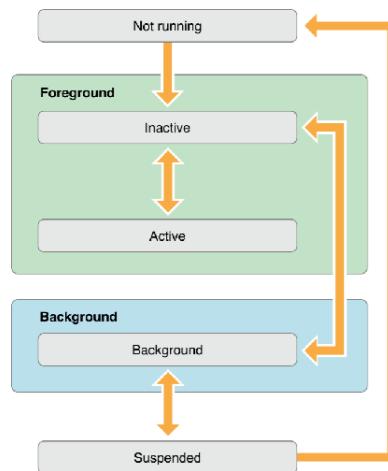
Cycle de vie des applications

39



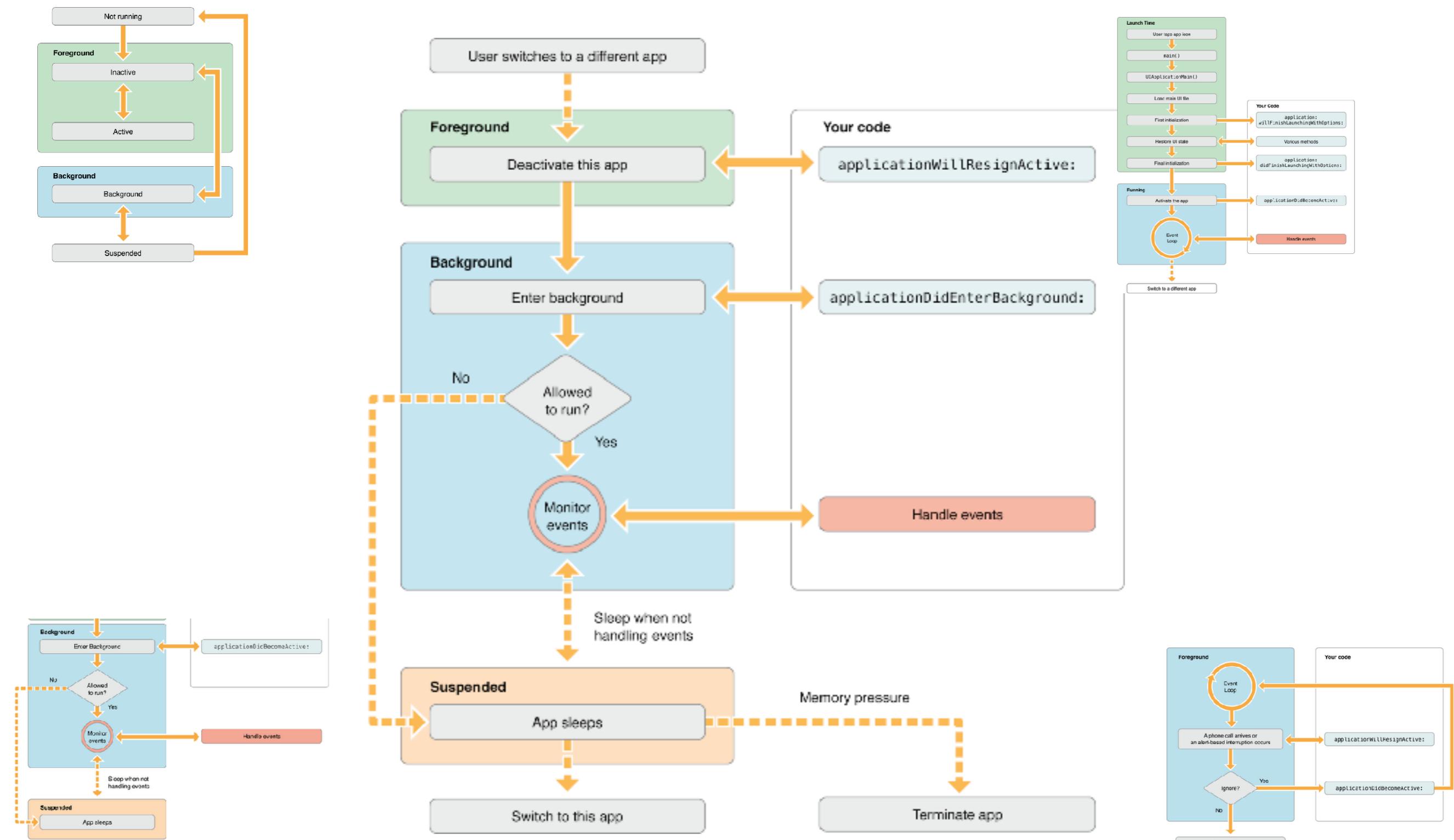
Cycle de vie des applications

39



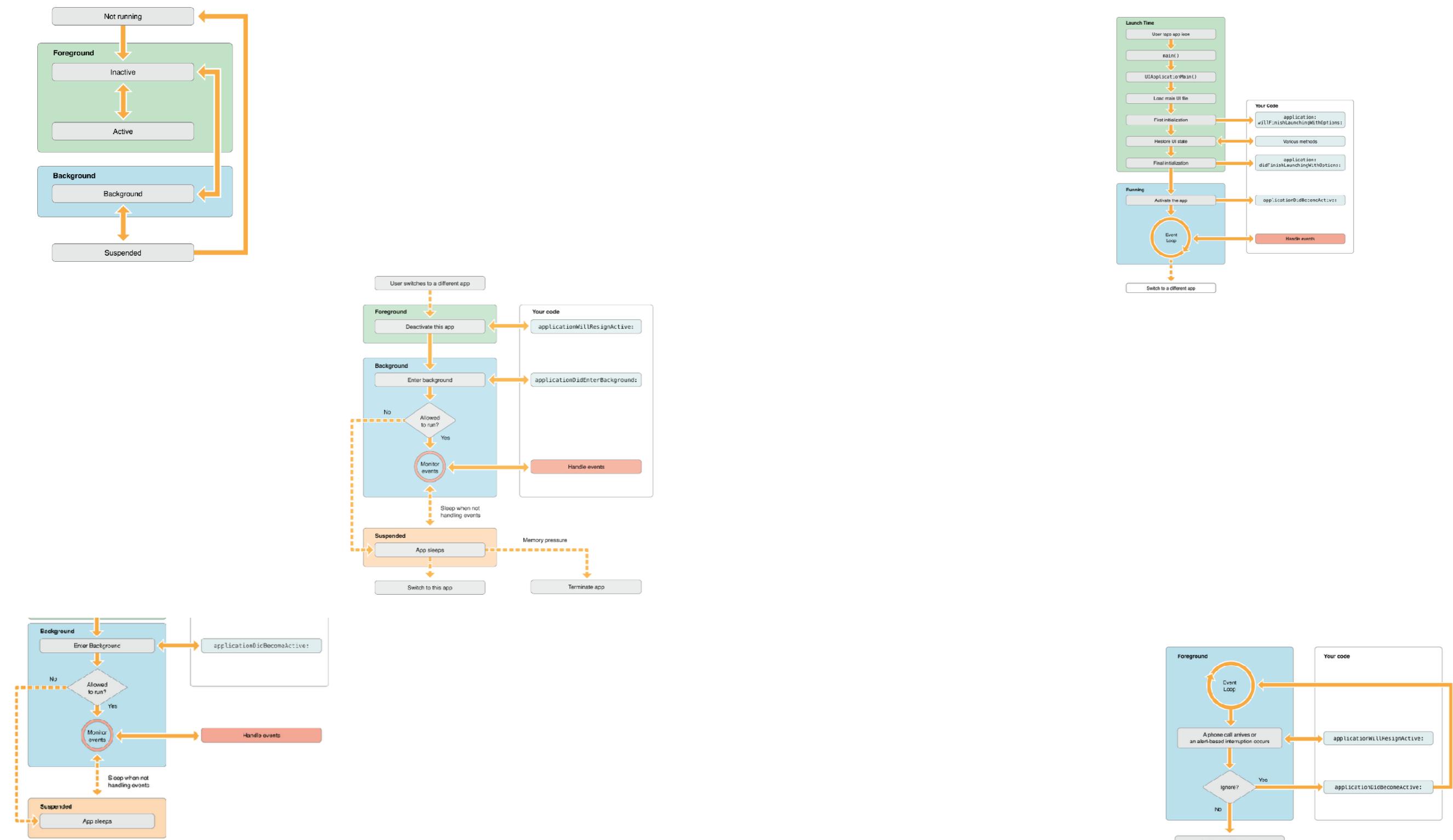
Cycle de vie des applications

39



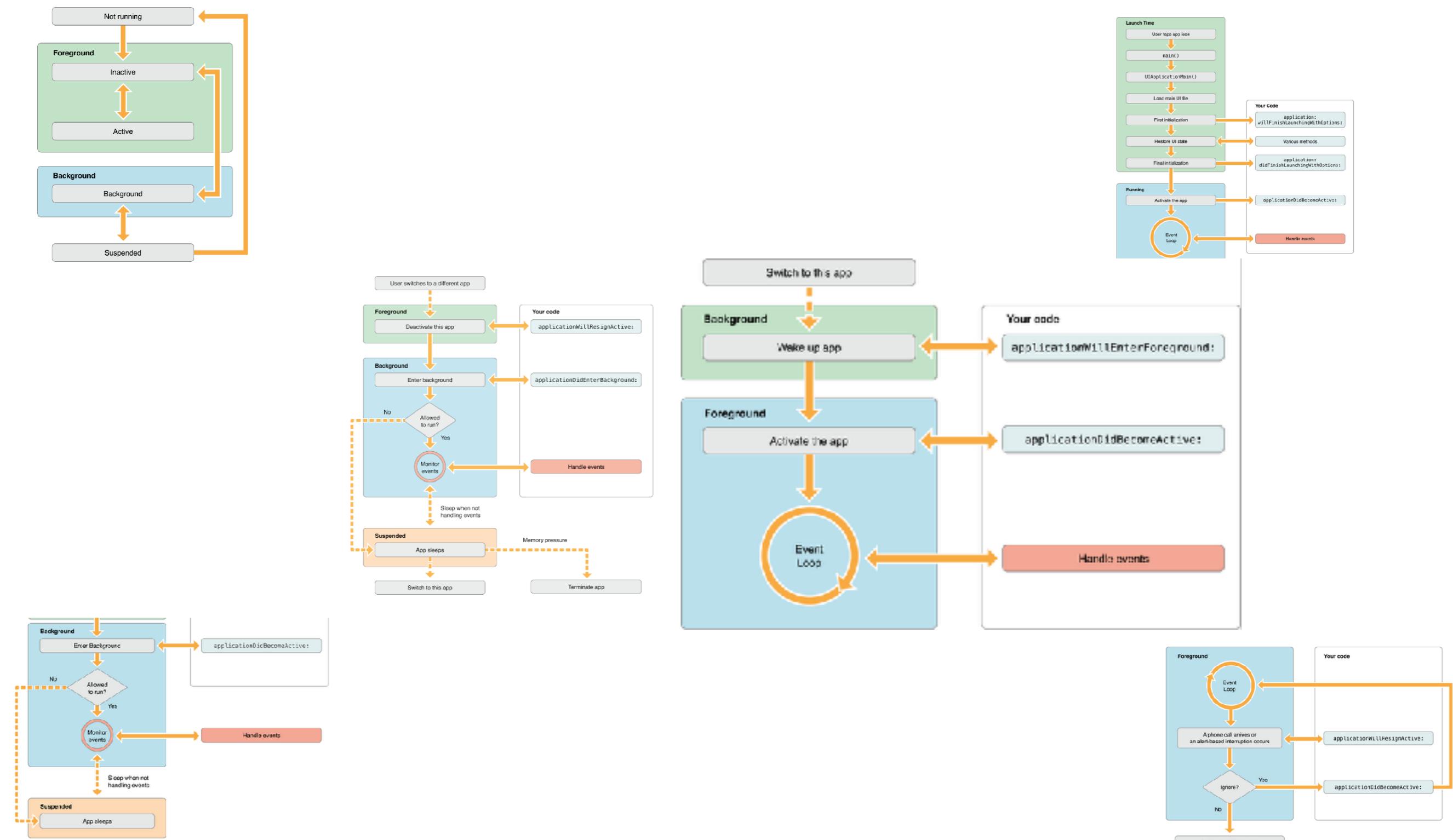
Cycle de vie des applications

39



Cycle de vie des applications

39



Programmation concurrente

40

```
1 int main(int argc, char *argv[ ])
2 {
3     @autoreleasepool {
4         return UIApplicationMain(argc, argv, nil
5                             , NSStringFromClass([NFA022_AppDelegate class]));
6     }
}
```

Programmation concurrente

40

- Le main thread est transformé en EventLooper => il faut exécuter les tâche longues sur des threads dédiées. IOS propose plusieurs outils classiques :
 - NSThreads
 - NSTimers
 - NSOperations
- L'apparition de systèmes multi cœurs CPU et la monté en puissance de GPU embarqués laisse entrevoir de nouvelles possibilités (Accelerator, Metal APIs...)

Modèle de sécurité

Modèle de sécurité

41

- Le Sandboxing applicatif pris en charge par iOS interdit pratiquement tout échange de données entre application. Les possibilités se limitent en générale à :

Modèle de sécurité

41

- Le Sandboxing applicatif pris en charge par iOS interdit pratiquement tout échange de données entre application. Les possibilités se limitent en générale à :
 - la recopie de fichier d'une application vers une autre
 - l'échange de données par mécanisme du copier/coller depuis la version 3.0
 - l'échange à travers le réseau (messagerie, cloud ...)
- Toutes les ressources d'une application installée sur un terminal sont signées et le noyau iOS vérifie ces signatures à chaque chargement. L'OS cantonne les accès d'une application à son stricte répertoire de travail (le UserDomain) grâce à un service spécifique

Modèle de sécurité

42

- Ce modèle de sécurité très frustrant en termes de fonctionnalités :
 - Rassure sur les aspects sécurité les utilisateurs (contrôles peu contournables). Modèle de sécurité de l'AppStore basé sur la confiance vis à vis du constructeur et de sa boutique -> tests de compatibilité et de sécurité réalisés par Apple.
 - Mais est contradictoire avec le dogme une application : une tâche car pas d'interopérabilité entre applications, les applications doivent implanter toutes les fonctionnalités et réinventer la roue pour disposer d'un processus complet.

Les iOS Dev-Programs

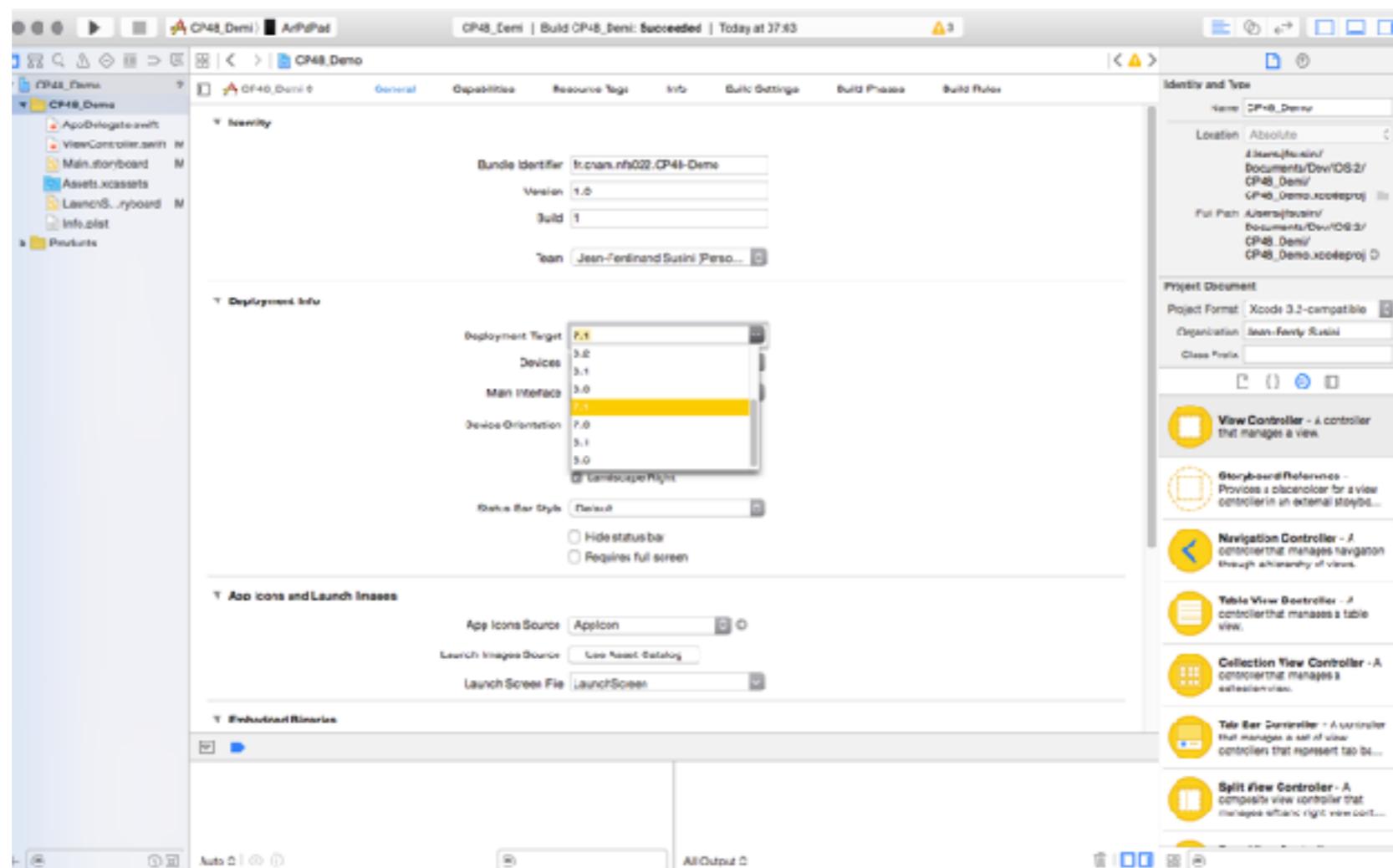
43

- Développer pour iOS, signifie développer sur MacOS X avec l'environnement Xcode. L'environnement est téléchargeable gratuitement. (Registered Developer)

Les iOS Dev-Programs

43

- Développer pour iOS, signifie développer sur MacOS X avec l'environnement Xcode. L'environnement est téléchargeable gratuitement. (Registered Developer)



Les iOS Dev-Programs

43

- Développer pour iOS, signifie développer sur MacOS X avec l'environnement Xcode. L'environnement est téléchargeable gratuitement (Registered Developer)
Depuis Xcode 7 ceci n'est plus vrai !
- Le développement avec test sur simulateur est libre. ~~Mais il est impossible d'installer les programmes sur un appareil.~~

Les iOS Dev-Programs

43

- Développer pour iOS, signifie développer sur MacOS X avec l'environnement Xcode. L'environnement est ~~téléchargeable gratuitement (Registered Developer)~~
Depuis Xcode 7 ceci n'est plus vrai !
- Le développement avec test sur simulateur est libre.
~~Mais il est impossible d'installer les programmes sur un appareil.~~
Depuis Xcode 7 ceci n'est plus vrai !
- iOS Developer Program : payant, permet d'obtenir des certificats pour pouvoir tester sur des appareils enregistrés (un certificat par appareil enregistré).
Enregistrement nécessaire pour un déploiement sur AppStore.

Les iOS Dev-Programs

43

- Développer pour iOS, signifie développer sur MacOS X avec l'environnement Xcode. L'environnement est téléchargeable gratuitement. (Registered Developer)
- Le développement avec test sur simulateur est libre. ~~Mais il est impossible d'installer les programmes sur un appareil.~~
- iOS Developer Program : payant, permet d'obtenir des certificats pour pouvoir tester sur des appareils enregistrés (un certificat par appareil enregistré). Enregistrement nécessaire pour un déploiement sur AppStore.